Learning Programming Using Python: The Case of the DigiWorld Educational Game

O. Dallas and A. Gogoulou

***Abstract* — Constructivist approaches have been demonstrated to contribute effectively to learning programming concepts. Game-based learning offers a novel and productive constructivist approach in this direction. Unfortunately, there is a lack of available educational games in the Greek language for teaching and learning programming using a programming language, especially Python. This paper describes the design and implementation of an educational game designed to teach programming through Python at an introductory level in the context of Greek upper secondary education. The game uses the Ren'Py visual novel creation toolbox, digital storytelling techniques and focuses on the narrative and the basic tenets of game-based learning to create a state of flow instead of simple gamification techniques. The outcome is a fully developed educational game, titled "DigiWorld", which can be used in the classroom as an educational tool, as well as in non-formal and informal education. The results of the preliminary evaluation are encouraging regarding the motivation and user's engagement as well as the educational approach followed through the challenges and feats designed.**

***Keywords* — educational games, game-based learning, Python, secondary education, learning programming.**

I. Introduction

The employment of constructivist approaches contributes effectively to learning generally, and in learning programming concepts specifically, as demonstrated by multiple studies and research work [1]-[3], [15], [19]. Game-based learning is a type of experiential learning based on the experience provided by a game, in which the player is active instead of being a passive recipient of knowledge [4]. Basic characteristics of a game, as defined by Plass *et al*. [5] include:

- the challenge, usually the task to be completed, the choice to be made or the difficulty to be overcome, depending on the nature of the game;
- the response made by the player, which includes any actions, choices or decisions as deemed appropriate by the player to succeed;
- the feedback provided, describing the outcome and success or failure of the action, along with any extra information that the game is designed to offer, such as a score or percentage of completion.

It is the last part, the feedback, which can provide an invaluable experience which can later be analyzed by an educator or facilitator, or simply be used to acquire new skills and knowledge or practice existing ones.

Game-based learning is different from gamification, as the latter adapts secondary elements of gaming such as scoring systems, badges, and awards in order to create extrinsic motivation to perform mundane or boring tasks such as homework. Instead, game-based learning focuses on creating fully functional games which teach through the experience of playing the game, aiming to create intrinsic motivations and achieve a state of flow, where the player learns without even realizing it by focusing on his or her act of playing.

Digital Storytelling is an art form which involves telling stories, an archetypal human activity, through digital media, mainly pictures, video, animation, and sound [16]. There exists, however, a particular type of game which uses elements of Digital Storytelling to provide a gaming experience: Visual Novels. Visual novels are a predominantly Japanese genre of narratively driven games with an emphasis on player choices [6]. They utilize text and static imagery, mostly, to tell a story in which the protagonist is the player, making choices and guiding the plot according to these choices and the events happening in the game. Visual Novels offer the capability to create a strong, compelling narrative able to create intrinsic motivation to the players to finish it, while offering the chance to insert educational material organically as long as it is tied to the narrative, so as to avoid ludonarrative dissonance [7].

When it comes to learning programming concepts and specifically through the Python programming language, the available games are limited and moreover present flaws concerning their introduction into a typical greek educational context. A study of the available literature offers only limited possibilities: Prog & Play [1], a real-time strategy game with limited accessibility due to requiring very specific infrastructure, and CodeCombat [10] and Ozaria [11], both from the same company, which even though brilliant and also available in the Greek language, they are subscription-based software aimed at schools, which presents an additional set of problems for using them as tools in the classroom.

The work presented, focuses on the design of an educational game, called DigiWorld which aims to support learning programming at an introductory level through the Python language. In the next section, a review of the field of educational games for learning programming, and in particular python, are briefly presented. Next, the design principles of the DigiWorld game are described followed by a discussion of the results of the pilot evaluation.

Submitted on November 4, 2021.
Published on February 16, 2022.
O. Dallas, National & Kapodistrian University of Athens, Greece.
(e-mail: odysseasdallas@gmail.com)
A. Gogoulou, National & Kapodistrian University of Athens, Greece.
(e-mail: rgog@di.uoa.gr)

II. Educational Games for Learning Programming

While there are a lot of games and other gamified activities which are available for learning programming [3], [8], [9], [17] very few actually are specifically for Python: only the following three were found at the time of this writing: Prog&Play, CodeCombat and Ozaria, as mentioned above. These games were evaluated based on a set of criteria devised by Gibson [9, pp. 40-53], as originally seen in [12], with the purpose of assessing their capabilities as tools for teaching the Python programming language in Greek secondary education.

The following criteria were used:

1. What is the subject?
2. How effectively is the subject taught?
3. How does it teach the chosen subject?
4. Availability

Other games that were evaluated were the Binary Headloom Game, Catacombs, Lightbot 2.0, Talent and Wu's Castle, among others, but aside from the aforementioned three, none were appropriate to support the teaching and learning process of the Python language. Out of a total of twenty five games, most were actually found to teach either basic programming concepts using visual programming (CargoBot, Robozzle) or simple movement commands (Lightbot, Lightbot 2.0), or a microlanguage (Catacombs, Saving Sera) specific to that particular game or environment.

Another important mention is gamified environments focusing on teaching programming, such as CheckIO, Codecademy, CodeWars and CodeZen. Even though such environments often included gamified aspects such as score, leaderboards, or badges, they are not actual games. Some skirt the lines by including a more coherent narrative, such as CheckIO which includes "missions" in different "islands". Unfortunately, gamification is quite different from game-based learning, as the former relies on a few aspects of games to create extrinsic motivation for players to continue playing, while the latter uses games (educational or recreational) to teach specific material through the act of playing the game, while at the same time creating intrinsic motivations in players who genuinely wish to keep playing for the sake of the game itself instead of simple scores or achievements [18].

A. Prog & Play

The first game evaluated was Prog&Play, by Muratet *et al*. [1]. It is an educational real-time strategy game, where each player controls a selection of units by coding their orders in one of the supported programming languages, such as C++ and Python. The narrative of the game follows a scenario where a hidden battle rages inside a computer, and the "army" of each player is composed of "bits", "bytes", "assemblers" and other units with a futuristic aesthetic reminiscent of science fiction movies.

1. The subject is teaching computer programming in a variety of languages, including Python.
2. The game teaches actively, as the players code the instructions of the units in the accompanying programming interface, and see the outcome in the level played, receiving the appropriate feedback.
3. It is also designed as a classroom tool, to be used after teaching the necessary theory, so it can be used as part of the exercising process. The game itself is a means of experimentation, offering a constructivist way to teach programming, as the players are allowed to code whatever solution they feel appropriate and try different approaches to solve the same problem using both previous knowledge and skills acquired through experimenting with coding.
4. The game is free, but requires extensive installation, both for itself and any supported languages. It is quite hard to use, requiring multiple simultaneous applications (the main application as well as a secondary interface where code is loaded after being written in a programming editor or other similar application), and finally it is available only in English and French.

B. CodeCombat

The second game evaluated was CodeCombat [10], a product of CodeCombat Inc., in which a player controls a character by writing instructions on a text editor embedded in the webpage. The results of the instructions are visible as the outcome of the level shown on the left, while the character navigates the level, fights off enemies and achieves various objectives. There is no particular plot in this game, merely a collection of levels.

1. The subject is teaching computer programming in either Python or JavaScript.
2. The game teaches actively, as the players code the solution to each level and see the results immediately, receiving the appropriate feedback.
3. The knowledge taught is mostly procedural and secondarily strategic, learned in a constructivist way by having main and secondary objectives intended to practice and automate programming skills through continuous coding.
4. The game is browser-based and available in many languages, including Greek, but requires a school-based subscription to be available with its full plethora of levels and supporting tools, making price its main obstacle to being available.

C. Ozaria

The third and final game considered was Ozaria [11], also a product of CodeCombat Inc., and its successor. As before, a player controls a character, this time a specific one tied to the narrative, by writing instructions on a text editor embedded in the webpage; on the left, the outcome of those instructions becomes visible, as the character navigates the level to achieve various objectives. Unlike CodeCombat, there is a very strong thematic narrative permeating the entirety of the game, including voice-over cinematic cutscenes, and there's a decrease of combat as well.

1. The subject is teaching computer programming in either Python or JavaScript.
2. The game teaches actively, as the players control their character through coding and receive the appropriate feedback immediately, while utilizing the available narrative create a state of "flow".

3. While there is a concise presentation of theoretical programming terms to increase declarative knowledge, the focus of the game is in practicing and automating programming skills through a constructivist approach of specific learning objectives-oriented levels which require a programming solution.
4. As in CodeCombat, the game is browser-based and available in Greek as well as other languages. As it is also a product, it requires a school-based subscription to be fully available along with all the content and the support tools that enable it to be used as a classroom tool.

D. Conclusions

After evaluating all three games, certain patterns began to emerge: all games promote students' active involvement in learning programming, through a constructivist approach and in a dynamic way, focusing on procedural and strategic instead of declarative knowledge, in order to develop programming skills.

However, these games had a major issue as far as availability is concerned; the first one, is not available in Greek and presented problems in both installing and running it, while the other two were limited by the necessity of a school-wide subscription. Furthermore, with the exception of Ozaria, neither of the other two provided a particularly immersive narrative, let alone one crafted as a product of Digital Storytelling principles. Last but not least, in all three cases, there was no particular challenge tying the programming process with the story; it is either a given in the game (as in Prog & Play or CodeCombat) or considered a kind of "magic" (as in Ozaria), yet in all three cases the programming is merely the control of the character or unit(s) in the level, with an occasional alteration of the environment because of the code.

Therefore, in the direction of creating and providing a proper classroom tool for learning programming through Python at an introductory level, it is necessary to provide a game with a strong narrative and constructivist approach to develop knowledge and skills at various levels, and be available in Greek, easy to install and use, and free- otherwise, it will simply not be used in a classroom, failing its purpose.

III. The DigiWorld Educational Game

A. Design Principles

The design of the DigiWorld game was based on fundamental game elements and the four axes, as defined by Plass *et al.* [5], in combination with the CMX design framework as presented in Malliarakis *et al*. [13].

The design of an educational game provides unique challenges not commonly encountered in conventional game design theory [14]. Thus, it is important to adopt a proper design framework, and in this case the CMX design framework for designing educational games to teach computer programming was used [13]. This particular framework was designed by Malliarakis *et al*. [13] with a very specific philosophy: to teach computer programming using video games, in particular massively multiplayer online role playing games (MMORPGs), but also applicable to any kind of educational computer game. While several other frameworks are available [13], such as the Four-Dimensional framework by de Freitas & Jarvis or the Model for Educational Game Design by Song & Zhang, CMX was deemed the most appropriate not only because it was created by taking the above, among others, into consideration, but also because it was specifically designed as a framework for designing computer games that teach programming skills for secondary school students.

Case in point, Fig. 1 shows the CMX design Framework specifically for the DigiWorld game, which takes the form of a Mind Map. Six main, inter-related topics are the basis of the framework:

- Learning Objectives, which codify the learning objectives that are necessary for the proper educational design of the game; the main ones involve combining recreation and education to create python programs and solve problems using Python, both individually and in groups.
- Pedagogy, which analyzes the teaching-learning model and the organization of the educational material according to the general time frame, the specific learning content, and the unit of learning. In DigiWorld, emphasis is given in problem-solving activities. The students are getting involved in various problem solving activities during the learning phase as well as afterwards in order to apply their knowledge. The unit of learning is the Challenge; each programming challenge has specific programming concepts or commands to teach, and it comes complete with theory, help and its own problem to solve. A breakdown of the challenges and the corresponding educational content is presented in Table I.
- Learning Outcomes, which prescribe the specific learning outcomes achieved by playing the game. In the framework above we see a general approach in the levels of understanding and applying; the DigiWorld game has specific learning outcomes for each different challenge addressing both levels. Also, the game aims at developing ICT skills through the interaction with the scaffolding and help facilities as well as critical thinking and reflection through the critical study and use of the provided feedback.
- Users, which categorizes the different users which will be interacting with the game's infrastructure and the game itself; in the case of DigiWorld, only students/players and the educators.
- Scenario, which describes the narrative context as well as the plot of the game, its characters, and the goals to be achieved. In DigiWorld, the plot is divided into three acts, while the narrative is placed in a future dystopian world where programming is as skill restricted only to government employees- and therefore, the student protagonist practices it under the guidance of his mentor Armodios [8].

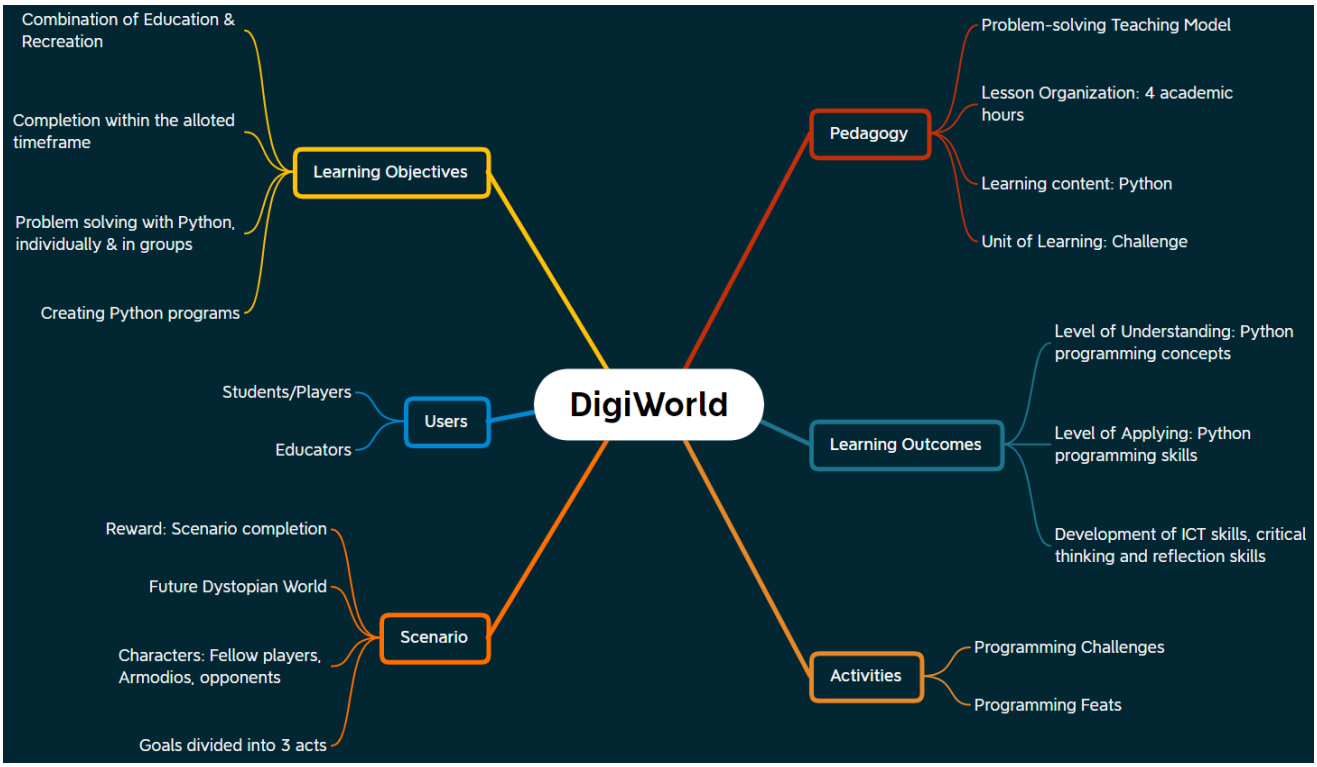

Fig. 1. The CMX Design Framework for the DigiWorld game.

TABLE I: THE PROGRAMMING CHALLENGES IN DIGIWORLD

Act	Challenge	Programming Concepts/Commands
I	1st (A-B-C)	Program output, "print" command, strings
	2nd	Variables, assigning values to variables, value assignment operator "="
II	3rd	Operations on variables, addition operator "+", multiplication operator "*"
	4th	Handling strings, concatenating strings, concatenation operator "+"
	5th	Boolean values, assigning a Boolean value to a variable
	6th	Conditional statements, "if" statement
	7th	Conditional statements," if-else" statement
	8th	Conditional statements," if-elif-else" statement
	9th (A-B)	Objects, object properties, assigning value to a variable which is an object property
	10th	Iteration statements, "for" loop
	11th	Iteration statements, "while" loop
	12th	Iteration statements, "break" command
	13th	Functions without parameters, calling functions without parameters
	14th	Iteration statements, nested loops
	15th	Functions with parameters, calling functions with parameters
	16th	Calling functions which are object properties, calling functions with parameters which are object properties
III	17th	Functions which output results, defining functions which output results
	18th	Lists, implementing algorithm to find shortest distance between points in a list, using "for" loop to loop over lists
	19th	Lists, initializing a list, adding list elements under conditions
	20th	Programming Feat

- Activities, which delineates the ways that players interact with the game and take actions in it, as well as the ways the expected learning outcomes are achieved. In DigiWorld, following a problem-solving based approach, programming challenges and feats are available, which involve the interactive choices the players take within the narrative that result in the continuation of the plot in different ways as well as the solving of various problems by writing programs in Python.

In combination with the CMX design framework, the principles of game-based learning [5] as well as digital storytelling techniques were used to provide a wholesome approach to implementing DigiWorld, so as to include educational material in the form of Python coding challenges without creating the impression they are out of place or inappropriate for the context of the game. Specifically, DigiWorld was designed along four main axes of design:

- Cognitive, the knowledge and skills that the game teaches, in this particular case theoretical and practical knowledge of programming in Python, with specific

learning objectives and learning outcomes achieved through the activities of programming challenges and feats, as seen above.

- Motivational, which covers both intrinsic, like the scenario and the gameplay, and extrinsic, such as leaderboards and badges, motivational elements. In the case of DigiWorld, the latter is absent by design; the value of extrinsic motivation has not been fully proven yet, while intrinsic motivation is achieved through immersion in the narrative and the scenario.
- Affective, dealing with the emotions and feelings caused by the game as well as behaviors and attitudes involving the game. Specifically, the very skill of programming in DigiWorld is considered a rare and powerful capability that is restricted to government agents, due to the power it gives to its wielders in the almost completely digitalized world of the future, as presented within the scenario. Therefore, by exaggerating its importance and value, it is possible for students (secondary education students who will soon be choosing which studies to follow) to become more positive towards Computer Science in general and programming in particular.
- Sociocultural, which includes not only cultural elements which influence both the creation and the playing of the game, but also the entire external framework and its social aspects. The classroom and the school is the sociocultural environment that DigiWorld was designed for, from the scenario and characters down to the dialogue and form of speech. Even more so, the plot of the game deals with issues of dystopia, authoritarianism and rebellion, themes which feature prominently in young adult literature and even everyday lives.

Furthermore, the educational content of the game was designed with the appropriate learning outcomes in mind, following a natural progression of knowledge and skill acquisition which begins with basic programming concepts, such as variables, conditions and loops, and their implementation in Python, and ends with problem-solving using coding techniques of the Python language. The mentality of the game is based on the seamless combination of the recreational with the educational aspect, under the premise that a game which does not entertain is not played, therefore it does not educate.

B. Functionality

DigiWorld offers two different modes of play: the Quick game and the Full game. The Quick game follows a rather simplified and linear plot, limiting the amount of text and extra choices so that the player can focus on the programming challenges in the game. These limitations respectively limit the amount of time variance in finishing the game depending on a player's reading speed, making the Quick game ideal to use in the time-limited context of a classroom, offering a full game that does not sacrifice any educational material for the sake of playing.

The Full game, on the other hand, has the same programming challenges as the Quick game, but introduces a lot more narrative, additional choices which shape the plot to an extent, and extra scenes to play through. Its main goal is, time notwithstanding, to offer a more complete experience which can more easily create a state of "flow" in the students by immersing them in atmosphere and narrative of DigiWorld's fictional environment.

Furthermore, it is important to note that while in both versions of the game the same set of programming challenges are encountered (as presented in Table 1), the last programming challenge to complete the game is a Programming Feat. These Feats are much harder than the other challenges, requiring the student to solve a problem using Python. A list of twelve Programming Feats have been implemented- each time, one is randomly chosen as the final challenge. Also, students have access to all twelve from the Main Menu and can try to solve them at any point as extra activities if they wish.

On Table I, the programming challenges in DigiWorld are presented in order of their encounter in the game, along with the programming concepts and commands they teach using Python.

The plot of DigiWorld is structured in three very specific story acts; the first part begins with the protagonist in the school eager to play a mission in their favorite online game and ends with the success of the mission. The second part, which also has the majority of the programming challenges, including raising a baby dragon (as shown in Fig. 2), begins right after the protagonist returns from the mission and starts becoming more popular, until betrayed by the faction leader to the government. The third part has the protagonist run to safety, using his/her programming skills in the "real" world instead of the digital one, until s/he manages to hack a government database and uncover incriminating evidence, which s/he can use to either spark a riot or bargain for immunity (player's choice).

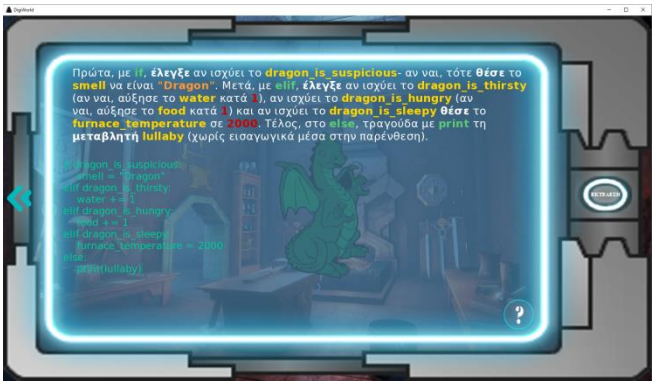

Fig. 2. Programming Screen for the eighth programming challenge, where the player is raising a dragon using if-elif-else commands. The dragon is crucial for the success of the protagonist's endeavors.

Fig. 3: Outcome screen for the first programming feat.

Finally, since DigiWorld was created to be a tool for classroom use in secondary education, it comes with a complete lesson plan, as well as suggestions for using it as homework or even as a tool for self-learning. The lesson plan is organized in four academic hours of 45 minutes each. Each academic hour combines playing the game with reflection time on the specific concepts taught, to maximize learning gains; the first hour also includes 10 minutes to familiarize the players with the game as well. The teacher also has the capability of assigning challenges as homework, which s/he can check either by the students sending screenshots directly taken from the game (an available feature) or even saved games with their progress.

C. Evaluation

After the completion of the DigiWorld game, the evaluation was done in two ways: one was to evaluate it according to the same set of criteria used to evaluate the games in section II, and the second was to perform a practical evaluation through an experimental application with a sample of the students attending the Educational & Teaching Competency Program at the Department of Informatics & Telecommunications (National & Kapodistrian University of Athens).

Initially, the game was evaluated by the authors according to the aforementioned criteria (Section II) which were also used for the evaluation of the three games focusing on the Python language:

1. The subject is teaching computer programming and basic programming concepts in Python.
2. The game activates students, as the players face the programming challenges by writing code, which is dynamically run within the program, returning the appropriate feedback. The narrative of the game is a vital factor that leads in immersion and achieving a state of "flow".
3. DigiWorld focuses on three different levels of learning, when it comes to learning programming through Python: acquiring and practicing programming skills, learning and understanding basic programming concepts through theoretical presentations embedded in the narrative (see Fig. 4, where the mentor character Armodios is explaining the assignment of values to variables using the "=" operator, right before the character is about to fight a dragon), and providing a different way to look at programming not as a simple technical skill but a powerful competency that affects the world, akin to a superpower of sorts.
4. First and foremost, since DigiWorld is a product of academic research, it is freely available. It is also easy to download, install (simply unzip the file) and play, without requiring administration privileges or even internet access (after the download). Finally, it comes with additional support material for teachers, to aid in integrating it with the curriculum of secondary education.

Fig. 4. The mentor character Armodios explaining theory within the game.

The experimental application took place over teleconference, considering the health regulations regarding the pandemic. A total of 25 people took part in it, for a duration of three hours; about one and a half to two hours to play the game, and the rest of the time used for discussion. The participants were chosen because they could contribute to the research by virtue of having both student and (potentially) teacher roles at the same time. While not being secondary education students themselves, they had the capacity of being educators in training with the potential to use DigiWorld as an educational tool in the future. The overwhelming majority was students aged 21 to 23. Three of them were Informatics teachers, one at primary level and three at secondary level. The participants' background in Python is depicted in Fig. 5 showing that the majority had little experience in Python.

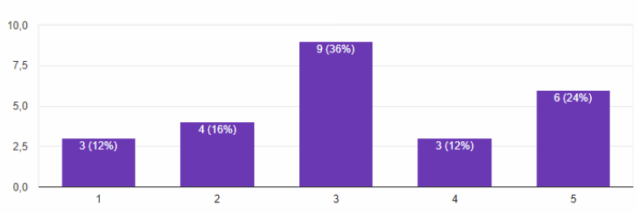

Fig 5. Sample background in Python.

The tools used during the experimental application were observation, discussion during the experiment, and completion of a survey.

The observation was unstructured, conducted by the authors, and resulted in certain insights regarding the positive reception of the game, mostly focusing on its narrative and visual elements as a way of motivating students to complete it. During the observation, the authors were working supportively, in the same manner as a teacher within the classroom. A key insight was that not only the speed of completion varied considerably and was linked to former Python background, but many also exhibited issues of mistyping due to backgrounds in other programming languages (such as C and the ubiquitous semi-colon, which caused syntax errors in Python) and over-dependence on auto-correcting programming editors (one third of the participants made spelling errors in commands or variable names, and half had issues with indentation).

The discussion that followed added the following conclusions: the game is highly motivating, both by nature and design to teach Python; some would like to see a different kind of game with the same educational content; a couple of students declared they would use it as an

additional tool, preferably for use at home, instead of a classroom tool, since they did not feel confident of how to effectively utilize it in classroom.

Finally, the survey offered a quantitative validation of the DigiWorld game. Indicative statistics include the following, as out of the 25 participants:

- 44% rated the programming challenges as moderately hard, though no one rated it as very easy or very hard.
- 76% rated the programming challenges as appropriate or very appropriate for the chosen learning outcomes.
- 84% rated the visualization of programming challenges as helpful or very helpful.
- 92% rated the programming challenges as adequately or fully connected to the narrative.
- 80% rated the help provided to the programming challenges as adequately or fully appropriate.
- 76% considered the programming challenges succeeded in their learning outcomes.
- 88% considered the programming challenges appropriate for secondary education.
- 64% rated programming feats as moderately hard requiring skills in problem-solving and in using Python language.
- 72% rated the visualization of programming feats as helpful or very helpful.
- 84% rated the help provided to the programming feats as adequately or fully appropriate.
- 72% rated programming feats as adequately or very interesting as an overall impression.
- 76% considered the programming feats succeeded in their learning outcomes.
- 80% considered the programming feats appropriate for secondary education.
- 92% considered the visual elements of the game either pleasant and useful, or very pleasant and useful.
- 92% considered the narrative of the game either game-enhancing or very game-enhancing.
- 80% considered the activities implemented in the game (narrative progression and programming) to be fully adequate, and 20% to be partially adequate.
- 64% admitted to achieving a state of "flow" while playing, while 36% only partially or at times.
- 88% considered the educational design was implemented either successfully or very successfully.
- 92% answered that the game's overall learning outcomes were achieved either mostly or completely.

IV. Conclusion

DigiWorld was created to address the gap in available games for teaching programming through Python in secondary education in Greek language, in an attempt to offer practical solutions to the difficulties of implementing game-based learning methods in classroom. The game seems to engage students actively in the learning of programming through Python and provide meaningful challenges and feedback that develop programming knowledge and skills. Also, the blending of digital story telling with game based learning, creates feelings of recreation in parallel to learning.

In the future, the goal is to pilot test the game in real classroom settings and expand and improve upon DigiWorld, offering additional content and improved functionality such as group playing. In addition, continual support for DigiWorld is planned, in the sense of providing the necessary support to secondary education teachers who wish to use DigiWorld in their classrooms.

References

[1] Muratet M., Torguet P., Jessel J., and Viallet F. Towards a Serious Game to Help Students LearnComputer Programming. *International Journal of Computer Games Technology*, 2009; vol. 2009, Article ID 470590, 12 pages. https://doi.org/10.1155/2009/470590.

[2] Livovský J., and Porubän J. Learning object-oriented paradigm by playing computer games: Concepts first approach. *Open Computer Science*, 2014.;4(3):171–182. doi:10.2478/s13537-014-0209-2.

[3] Malliarakis C., Satratzemi M., and Xinogalos S. Educational Games for Teaching Computer Programming. Research on E-Learning and ICT in Education. in *Research on e-Learning and ICT in Education*, C., Karagiannidis, P. Politis, I. Karasavvidis, Ed. Springer, New York, NY, 2014. https://doi.org/10.1007/978-1-4614-6501-0_7.

[4] Perrotta C., Featherstone G., Aston H., and Houghton E. Game-based learning: Latest evidence and future directions" (NFER Research Programme: Innovation in Education) 2013. Slough: NFER. Available: https://www.nfer.ac.uk/publications/GAME01/GAME01.pdf.

[5] Plass J. L., Homer B. D., and Kinzer C. K. Foundations of game-based learning. *Educational Psychologist*, 2015;50(4):258-283.

[6] Øygardslia K., Charlotte W., and Shin J. The Educational Potential of Visual Novel Games: Principles for Design, *Replaying Japan*, 2020;2(2). Available: https://www.researchgate.net/publication/341380379_The_Educational_Potential_of_Visual_Novel_Games_Principles_for_Design_Replaying_Japan_Vol_2.

[7] Seraphine F. Ludonarrative Dissonance: Is Storytelling About Reaching Harmony? 2016. Available: https://www.academia.edu/28205876/Ludonarrative_Dissonance_Is_Storytelling_About_Reaching_Harmony.

[8] Maragos K. and Grigoriadou M. Towards the design of intelligent educational gaming systems. In Proc. AIED05 Workshop 5: Educational Games as Intelligent Learning Environments, 2005, pp. 35-38.

[9] Gibson B. I. Educational games for teaching computer science. M.S. Thesis, Department of Computer Science and Software, Engineering, University of Canterbury, 2013. Available: https://ir.canterbury.ac.nz/handle/10092/9239.

[10] CodeCombat. Coding games to learn Python and JavaScript. Available: https://codecombat.com/.

[11] CodeCombat. Computer science that captivates. Available: https://www.ozaria.com/.

[12] Salen K., Tekinbas K.S. and Zimmerman E. *The game design reader: A rules of play anthology*, 2006, Cambridge, MA: MIT Press.

[13] Malliarakis C., Satratzemi M., and Xinogalos S. Designing Educational Games for Computer Programming: A Holistic Framework. *Electronic Journal of e-Learning*, 2014;12:281-298.

[14] Salen K., Tekinbas K.S., and Zimmerman E. *Rules of play: Game design fundamentals*, 2004. Cambridge, MA: The MIT Press.

[15] Papadakis S., and Kalogiannakis M. Evaluating the effectiveness of a game-based learning approach in modifying students' behavioural outcomes and competence, in an introductory programming course. A case study in Greece. *International Journal of Teaching and Case Studies*, 2019;10(3):235-250.

[16] Wu J. and Chen D.T.V. A systematic review of educational digital storytelling. *Computers & Education*, 2020;147:103786.

[17] Olsson M. and Mozelius P. Learning to Program by Playing Learning Games, *11th European Conference on Games Based Learning*, 2017;11:498-506, Graz, Austria, 2017.

[18] Karagiorgas D. N. and Niemann S. Gamification and game-based learning. *Journal of Educational Technology Systems*, 2017;45(4):499-519.

[19] Seralidou E. and Douligeris C. Learning programming by creating games through the use of structured activities in secondary education in Greece. *Education and Information Technologies*, 2021;26(1):859-898. https://doi.org/10.1007/s10639-020-10255-8.

O. Dallas was born in Athens, Greece, and received his degree on Computer Science from the Department of Informatics & Telecommunications of the National & Kapodistrian University of Athens in 2021.
He currently works as a Computer Science Teacher in primary education. He is also the co-founder and Game Developer of Dragons' Nest, a company in Athens, Greece specializing in using educational role-playing games as a tool for game-based learning non-formal education after-school activities. In the past, he has worked as a facilitator in non-formal education programs, including within the Erasmus+ framework. Currently, he is interested in game-based learning methodology and the development of educational games, both digital and off-line, such as role-playing games and board games.

A. Gogoulou is a member of the teaching staff of the Department of Informatics & Telecommunications, University of Athens in the subject of "Computer Science Education and ICT in Education". She has been a teacher in secondary education for twenty years and member of the Education and Language Technology Laboratory since 2002. She has rich teaching experience in both secondary and tertiary education and has been engaged in a number of adults training programs related to the use of ICT in various fields. She has also participated in the development of educational material for teachers' training programs and in committees for the design of informatics curricula for secondary education and for the evaluation of educational material. She has participated in research projects related to the introduction of new teaching and learning approaches and has supervised a number of theses in the field of learning design, game-based learning, e-learning, web-based learning environments. Her research interests are related to computer science education, learning design, game-based learning, peer-assessment, adaptive learning environments, e-learning. She has more than 40 publications in international journals and in proceedings of international conferences.