Olympiads in Informatics, 2016, Vol. 10, 39–60

DOI: 10.15388/ioi.2016.03

Learning Programming through Games and Contests: Overview, Characterisation and Discussion

Sébastien COMBÉFIS[1,2], Gytautas BERESNEVIČIUS[3]
Valentina DAGIENĖ[3]

[1] *Electronics and IT Unit, École Centrale des Arts et Métiers (ECAM)*
Promenade de l'Alma 50, 1200 Woluwé-Saint-Lambert, Belgium
[2] *Computer Science and IT in Education ASBL, Belgium*
[3] *Vilnius University Institute of Mathematics and Informatics*
4 Akademijos Street, Vilnius LT-08663, Lithuania
E-mails: s.combefis@ecam.be; gytber@gmail.com; valentina.dagiene@mii.vu.lt

Abstract. Learning of programming and, more generally, of computer science concepts is now reaching the public at large. It is not only reserved for people who studied informatics (computer science) or programming anymore. Teaching programming to schoolchildren presents many challenges: the big diversity in ability and aptitude levels; the big amount of different tools; the time-consuming nature of programming; and of course the difficulty to motivate schoolchildren to keep them busy with hard work. There are various platforms that offer to learn coding and programming, in particular game-based platforms, which are more and more popular. These latter exploits of the gamification process focused on increase in motivation and engagement of the learners. This paper reviews the main kinds of online platforms to learn programming and more general computer science concepts, and illustrates the review with concrete platforms examples.

Keywords: game-based learning, gamification, learning programming, online programming platform, programming contest.

1. Introduction

Informatics (computer science or computing) as a science discipline, and in particular programming as an important part of that, is gaining a lot of popularity these years in education. As core subjects in a computer science (CS) major, programming subjects play an important role in a successful CS education. One of the greatest challenges faced by most students is the understanding of programming basics, especially for novices or for those who are in their first year of studies. Students develop algorithmic

thinking or computational thinking at secondary and even at primary school. Using simple programming languages such as Scratch became accessible to young children (Maloney *et al.*, 2004).

Games have been used for educational purposes for decades. The popularity of games has led to the idea of using them in the learning of programming, taking advantage of the engaging features of games. In a very broad sense, two different approaches are used when using games or game-like elements in educational contexts: 1) gamification and 2) serious games. The term *gamification* is commonly defined as the use of game design elements in non-game contexts (Deterding *et al.*, 2011). Serious games are associated with a standalone game or platform, as well as indicated as a computer game (Djaout *et al.*, 2011). In this paper, we are going to use the general term to refer to game-based learning with main focus on teaching programming.

Game-based learning is concerned with using games not for entertainment, but for educational purposes. Those who work within the field of gamification focus on identifying the context and conditions that support the integration of digital games within informal and formal learning environments. Educational scientists have pointed up several features of games that allow them to be used as learning tools. For example, games are engaging (Dickey, 2005) and motivating (Prensky, 2003). They also provide a lot of experiences (Arena and Schwartz, 2013) and an excellent feedback on performances (Shute, 2011). Finally, games support very well the learner centred education (Gee, 2005).

A contest can be seen as a part of game-based learning. Contests make the teaching of programming more attractive for students. Furthermore, programming with computer is one of the appropriate and effective ways to develop problem solving skills and computational thinking. During contests, students have the possibility to compare their abilities and learn from others. There have been many contests in programming throughout all over the world; most of them focus on algorithmic problem solving.

Many teaching environments already contain game-like elements such as points, instant feedback, and goals. However, there are engaging aspects in games that could be used in educational settings more widely. In well-designed games, even a failure can be a reward and triggers positive emotions (Ravaja *et al.*, 2005). There is no off-the-shelf formula for designing successful games, nor is there one perfect learning environment. However, by deepening our understanding of the effects of games and game-like environments in educational settings, we can design and support more effective learning activities, and ultimately improve the learning of computer science.

The move from classical learning platforms to online contests and games can be explained as a way to ensure the best motivation as possible for their users. The online platforms can provide tools such as rankings, duels, discussion rooms, etc. to motivate their users to participate regularly. Finally, research on gamification in all its forms has proven that educational games improve the engagement of learners if the game is designed properly (Barata *et al.*, 2013; Nah *et al.*, 2014). Online platforms where students can learn programming are being developed all over the world, ranging from simple direct learning platforms to platforms of games that indirectly teach programming (Combéfis and Wautelet, 2014).

This paper reviews literature of the last decade on programming education for school students (high, secondary and primary levels) using games and online platforms that support games and contests which goal is to teach programming. The literature review of programming games and an overview of programming learning platforms were structured to accomplish the following three research objectives:

1. To overview the last decade literature on programming education using games.
2. To identify the important online platforms for learning programming through games and contests.
3. To identify suitable tools for programming novices.

The goal of the paper is twofold: from one side, to provide an overview of the work that has been done in this area, and from the other side, to discuss the important tools (online platforms and contests) and to improve programming education. After this introductory part, Section 2 makes a short presentation of the understanding of gamification and overviews the literature on teaching programming using games. Section 3 presents three kinds of online platforms, which allow students to learn programming. Then Section 4 tours several existing contest platforms. Finally, the last section concludes the paper with some thoughts about what could be the future evolution of these online platforms and contests and makes some suggestions for teaching programming by educational games or by other ways.

2. Short Overview of Literature

The application of game design elements of non-game scenarios is known as gamification. Some authors stated that gamification uses elements of experiences (Deterding *et al.*, 2011). It must also impart some rules or structure to the experience to influence participants' action. Gamification adds game design elements as well as promoting the psychological benefits and motivational ability of games but in different contexts.

Programming games are an important part of educational games. A programming game is a computer game where the player has little or no direct influence on the course of the game. Instead, a computer program or script is written in some domain-specific programming language in order to control the actions of the characters or other entities.

There are several definitions of serious games. For example, Alvarez and Michaud (2008) state that a serious game must include a genuine entertainment element combined seemingly with a practical dimension. Some researchers argue that all games have a serious purpose, such as gambling. That is why a majority of simulators would be considered as serious games (Sawyer and Rejeski, 2002).

Game design may be defined as *"the action of making sense of things related to a game"* (Mora *et al.*, 2015, p. 3). Another description is *"the act of deciding what a game should be"* (Schell, 2008, p. xxxviii). Game design is related to enjoyment, while gamification points towards a business objective. Similarity of game design and gamification design is that both rely on the principles of game design theory.

2.1. *Related Works*

The ACM and IEEE computer society state that *"computer science is becoming one of the core disciplines of a 21st century university education, that is, something that any educated individual must possess some level some level of proficiency and understanding of proficiency and understanding"* (ACM/IEEE…, 2013, p. 48). However, there are a lot of students who are becoming less interested in computer science. Colleges and universities routinely report that almost half of those students who initially choose computer science study soon decide to abandon it. There are high dropout rates as well as high failure rates of high school students in programming courses. Decades of effort have been put into decreasing these rates.

There were several approaches to motivate students to learn programming. Programming fundamentals may be hard skills to learn for students, especially for novices. In order to help students, there are several attempts to teach programming by using educational or serious games.

Programming courses are an important part in computer science subjects. Object-oriented programming is difficult for students, especially first year students. The embedding of gamification in programming courses could maximize student participation and have a positive impact on learning (Azmi *et al.*, 2015). The gamified social activity is designed to promote interaction among students and they may assist and give feedback to each other online. Online collaborative learning and the participation of students should be emphasised as well. Moreover, aesthetics is also a very important part in a game, because it makes it *"fun"*, which contributes to engage students in the learning process. The embedding of gamification in programming courses were analysed and showed that gamification in online collaborative learning can enhance participation among first year programming students. The participation is an important part in each computer science study due to motivation and lower dropout rates.

Vihavainen *et al.* (2014) present another approach in a literature overview. Their paper reports about 60 pre-interventions and post-interventions and analyses their influence on students' pass rates in programming courses. Statistically, pass rates after these interventions increased nearly by one third compared to traditional teaching. A more detailed discussion is presented about seven courses on gamification intervention after which pass rates increased by 10.8% on average.

Tillmann *et al.* (2013) deals with an alternative platform called *"Pex4Fun"*. On this platform, grading of traditional MOOCs (*Massive Open Online Course*) assignments has been changed to an automated grading assignment system based on the symbolic execution, designed for students and teachers. This platform is a game-based learning and teaching environment, where teachers may create interactive games for students. Students can learn programming by playing programming games in *"Pex4Fun"* as well as having programming duels between each other.

There are several important components of game elements. As Hunicke *et al.* (2004) have discovered the game mechanics represents data and algorithms. Game dynamics is the lifecycle of the mechanics that act to engage a player's input and other's output. Also, game aesthetics plays important role in the emotional part of the player in order to keep them engaged with the game.

Several researchers worked on more detailed classification of games. For example, Azmi *et al.* (2015) classified game elements into three categories 1) game mechanics (badges, leaderboard, points, levels...), 2) dynamics (personal dynamics – desire for reward, personal promotion; social dynamics – altruism; achievements, peer collaboration), and 3) aesthetics (challenges). The gamified design needs to embed collaborative elements for collaboration among the students. Aesthetics is recommended element in educational games, because it makes the game *"fun"* and engages with the learning process. Online educational game should have the online support as well as collaborative learning.

While there is still scepticism that something called a game could be anything more than a leisure activity, serious organizations are getting serious results with serious games. Serious games may be defined as an application with three main components: experience, entertainment, and multimedia (Laamarti *et al.*, 2014). Especially entertainment dimension in serious games has the potential to enhance the user's experience through multimodal interaction. Additionally, digital serious games contain different media.

The paper of Laamarti *et al.* (2014) overviews studies on serious games in different application areas including education, well-being, advertisement, cultural heritage, interpersonal communication, and health care. Also, many platforms were analysed and discussed, what the necessary components of the game to be successful are. A survey on digital serious games in general is conducted and a taxonomy of serious games is created.

Several components of serious games were distinguished and categories of criteria were suggested (Laamarti *et al.*, 2014). These criteria may be useful for any programming games as well. So, providing guidance to players within the game is important. This will provide them with the necessary knowledge and prevents them from feeling *"lost"* or disturbed. Another thing is avoiding negative consequences in the game, because otherwise it may result in the player's low performance. Music element makes players play the game longer, because of more motivation and engagement will be as an effect of this. Multiplayer collaborative exercise games are more motivating and engaging than single-player appropriate games. Collaborative environment increases players' motivation. It is significantly important to offer challenges in the game for children, but the challenges must be at the right level (neither too high, nor too low). It is the key in keeping the players' interest in the game. Educational games need to take into consideration sound instructional models to be successful.

More detailed recommendations are provided for educational games designed for using in a classroom. School teachers are concerned about the curriculum and they also have limited time resources. It is recommended that educational games would be based on the curriculum due to the acceptance of many teachers and integration in the class.

Additional recommendations for necessary components of the games to be successfully used can be as follows (Laamarti *et al.,* 2014):

a. **User-centred software engineering**: an important element is the perspective that the designers contribute to the development teams and the experience that the player will obtain.
b. **Multimodal games**: multiple modalities should be incorporated (e. g. visual, auditory and haptic combination).

c. **Social well-being**: stimulating a feeling of virtual presence or connectedness of social well-being in real life.
d. **Adaptive gaming**: a serious game should adapt or personalize a particular player's capabilities, needs, and interests.
e. **Standardization of evaluation**: heuristic evaluation standards must turn into a reality. That is useful for acquiring higher credibility.
f. **Sensory-based simulations**: serious games can be created based on real world sensory data so that real world scene would be accurately reconstructed.

Many commercial games demonstrate the properties of sound instructional design, even in games not intended for educational purposes (Becker, 2008). The author has recommended the following success factors for educational games: 1) Gain attention; 2) Inform learners of the objective; 3) Stimulate recall of prior learning; 4) Present stimulus material; 5) Provide learning guidance; 6) Elicit performance; 7) Provide feedback; 8) Assess performance; 9) Enhance retention and transfer. These factors were based on well-known learning and instructional theories.

One suggestion for recreating games is to modify an existing non-educational games into programming games (Muratet *et al.*, 2010). The authors have used existing online non-educational games ideas to develop their own serious games. For example, there is a game "*Kernel Panic*", having following features: it has no resource management except for time and space; all units are free to create; it has a small technology improvement tree with less than ten units; and it uses low-end vectorial graphics, which match the universe. In this game a player gives orders to the units to carry out operations (i.e. moving, building, and so forth). So the authors (Muratet *et al.*, 2010) changed the game design that the player has to write a program code in order to make the actions of his units and play. The authors modified some games into serious games and students, e.g. players may play these games using different programming languages: C, C++, Java, OCaml, Ada and an interpreted language called *"Compalgo"*.

Another paper (Cuba-Ricardo *et al.*, 2015) describes some characteristics and regularities from the behaviour of three students as they solved a programming problem as well as reveals the methodology stages using several methods, techniques and tools. The paper is limited by the time of contestants, who have between 60 and 90 minutes in contest of final states.

Successful computer programmers must have several cognitive skills as listed by Surakka and Malmi (2004). Contestants' opinions must be taken into account in order to have full view of their outputs. So, it was decided to use computer programs that take screenshots in short periods of time. The methodology of characterizing the cognitive process of solving programming problems was organized into five stages:

a. Preparation of the process.
b. Recording the process of exercise application.
c. Analysis, processing and assessing of the partial reports (observation of pictures).
d. An interview session.
e. Final assessment.

The implementation of the methodology revealed that programmers' took notes in the worksheets and the reasoning followed when determining the solution algorithm. So, monitoring programming students' actions is a useful method to discover not only final outputs, but intermediate results of the learning process as well. We think it may be one of the essential issues to be taken into consideration of teaching programming curriculum, especially for novices. Monitoring process could be done by computer automated programs and be used as a general method (e.g. it may be a live survey after your programming contest).

Researchers focus on investigating successful components of educational games that could help to reduce dropout rates of students in computer science. The embedding of gamification in programming courses can be one of solutions for that: it can help to maximize student participation and learning, motivate and reduce dropout rates, especially for novices in programming (Azmi *et al.*, 2015). Some platforms, e.g. *Prog&Play* may help even to recruit students in computer science (Muratet *et al.*, 2010). Intervention of educational games in programming courses raises pass rates by 10.8% on average (Vihavainen *et al.*, 2014).

A feedback is an important and necessary part in education and educational games (Azmi *et al.*, 2015; Cuba-Ricardo *et al.*, 2015; Tillmann *et al.*, 2013). Good feedback can be implemented by providing guidance to players within the game so that they do not feel *"lost"* (Becker, 2008) or by avoiding negative consequences in the game. Another solution is to make the game a multiplayer one to motivate players to play the game longer.

The most successful factors are multiple modalities of games, players' collaboration, adaptive or personalized game components based on real world sensory data (Laamarti *et al.*, 2014). Non-educational games may be gamified and used for teaching programming courses as well (Muratet *et al.*, 2010). Contests and duels in programming games are engaging and may be used to raise programming skills (Cuba-Ricardo *et al.*, 2015).

All the authors that are mentioned in this section have stated that educational games motivate students to learn. In general, you may use all these researched factors when your own create programming games.

2.2. *Taxonomy of Educational Games*

As there is growth of researches in the educational games field and it is a necessity to improve the value of teaching or learning programming by educational game, we were working on the taxonomy of educational games. We adapted the framework of serious games' taxonomy based on the paper (Laamarti *et al.*, 2014). Firstly, we will provide a short introduction to the mentioned taxonomy. After that we will discuss our novelties and provide a classification of several online programming games.

Five categories for serious games classification are presented by (Laamarti *et al.*, 2014, p. 6):

1. **Application Area** refers to different application domains (education, well-being, training, advertisement, interpersonal communication, health care, others).

2. **Activity.** It depends on the activity of a player to play a game (physical exertion, psychological, mental).
3. **Modality.** Here it is the channel by which information is sent from the computer to the human(s) participating in the game (visual, auditory, haptic, smell, others).
4. **Interaction Style.** The interaction style defines whether the interaction of the player with the game is done using keyboard, mouse, or Joystick or using some intelligent interfaces, e. g. a brain interface, eye gaze, movement tracking, and tangible interfaces. The research showed that choosing the right interface may have an impact on the success of the game.
5. **Environment.** This criterion defines the environment of the digital game (social presence, mixed reality, virtual environment, 2D/3D, local awareness, mobility and online).

We investigated the classification and found limitations, especially when applying for programming education by using games. For educational online programming games an extension and adaptation of the proposed taxonomy are needed (Fig. 1).

Application area. So, an application area of programming games is only one – education to program educational games. A field of the programming educational games activity usually is a mental process. We will have in mind that application area of educational programming games is education, activity field is mental and we will omit these fields as a result.

Modality. Usual modality fields of educational programming games are visual and auditory, but sometimes it may be haptic.

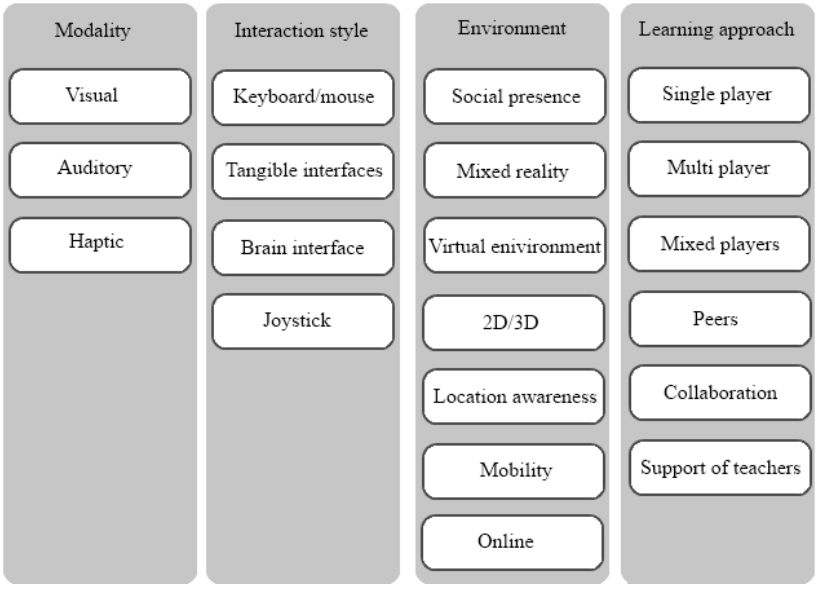

Fig.1. The classification of the educational programming games (adapted and extended schema of Laamarti *et al.*, 2014).

Interaction style. Interaction style in our taxonomy is keyboard/mouse, tangible interface, brain interface and joystick, because educational games usually may be classified just in these fields. But we remain open to discussion about this question.

Environment. As environments of programming games may be any of Laamarti *et al.* suggested environments, we include all of the areas: social presence, mixed reality, virtual environment, 2D/3D, local awareness, mobility and online.

Learning approach. We include this new field to classification of educational games. This is because analysed research papers on educational games pays a lot of attention to successful methods of teaching or learning. We propose that the learning approach of educational games may be classified into single player, multi player, mixed players (part of the game may be single-player game and part of the game may be multi-player game), peers (it is important to pay attention to communication between peers), collaboration (some educational games may teach team work), support of teachers (students easier learn while they are playing with a help of teachers/lecturers).

3. Online Platforms for Learning Programming

Several different kinds of online platforms to learn programming do exist. The most classical platforms are able to automatically execute code and provide feedback. They provide lessons that the learner can follow and propose interactive coding exercises with immediate correction. Combéfis and le Clément de Saint-Marcq (2012) proposes a short review of such platforms. Swacha and Baszuro (2013) propose such a classic platform, but they included game elements in it. For example, the courses in their platform are characterised by the set of completed areas, the number of earned points and the attained level. Moreover, it is possible to challenge other students and take part in contests.

In addition to the specific platforms to learn coding and programming, online courses are also proposed in the form of MOOCs (Combéfis *et al.*, 2014). Another example is Khan Academy, which provides interactive lessons and videos with coding exercises (Morrison and DiSalvo, 2014). Both these kinds of platforms propose a full course with videos for the theory and then practical coding exercises with more or less detailed direct automated feedback. They are designed with the learning process as the main objective.

This section presents three main categories of online game platforms with different goals: learn to code, learn algorithmic thinking and learn to create games. Proposed examples are discussed according to the background presented in the previous section, for the particular case of learning computer science skills. The last subsection, then discusses how and what game elements can be added to an online platform in order to gamify it.

3.1. *Learn to Code*

The first category of game platforms contains those whose goal is to make their users learning and training to code. Coding games require the learner to understand and to be

able to write code to solve challenges. Different coding activities are possible, the main ones being:

- **Bugfix**: the user is given a program that contains a bug to be fixed.
- **Recovery**: the user is given a program where some parts are missing that have to be filled.
- **Code writing**: the user has to write an instruction, a function or a program from scratch.
- **Agent**: the user has to write a program that represents the behaviour of an intelligent agent.

To help the learner to identify the bug, additional information is provided in the statement of the challenge. For example, a precise specification of the program or the results of the execution of test sets can be provided. Once the learner has written/fixed the code, he submits it and then gets a feedback. As summarised by Combéfis and Wautelet (2014), feedback is very important for the learning process. Those feedbacks can take several forms such as:

- A simple succeeded/failed status.
- The result of the execution of test sets.
- A textual feedback providing hints regarding the failure.

Codecademy is an example of an online game platform to learn how to code (Fig. 2). The left part of the window provides the explanations of the new concept to learn. Instructions about the exercise are then provided at the very bottom of this left pane. The learner can directly code in the editor located in the right part and then submit his/her code for correction. If the code is wrong, a simple feedback trying to explain why is provided and if it is right the learner earns a badge, increases his/her progress and can go to the next lesson.

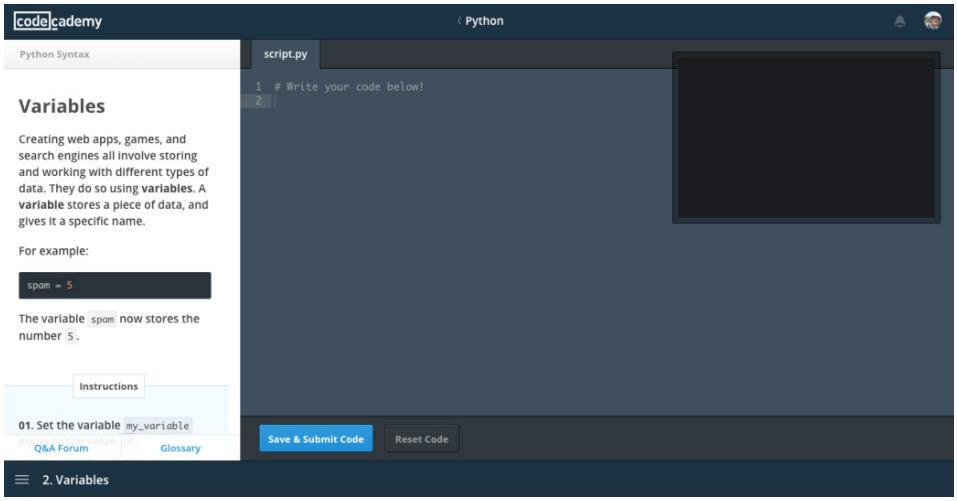

Fig. 2. *Codecademy* proposes a course composed of lessons in which you have to write code and for which you will earn badges and make progress whenever you succeed.

3.2. *Learn Algorithmic Thinking*

Coding is not the only field of computer science that can be taught through online platforms. As described by Combéfis *et al.* (2013), it is also possible to grow algorithmic thinking through interactive problems, in particular to learn programming concepts.

For example, *LightBot* (`https://lightbot.com`) is used to learn the notion of programming and in particular recursion through a simple game where the user has to solve puzzles. Each puzzle requires the learner to write a program composed of visual blocks to drive a robot to a goal. Fig. 3 shows the main window of the game where the robot and its environment are shown on the left and the program representing the behaviour of the robot is shown on the right. *LightBot* is not a platform to learn how to code, but to teach programming concepts. In the more advanced levels, the notions of function and therefore recursion are introduced.

Another example is *Initial Conditions* (`https://reheated.org/games/initial`), shown in Fig. 4, where the player has to find a solution to a search problem. In this game, the player faces to a grid with rivers and has to place a certain number of villages so that a river passes through them and so that no two cities are adjacent (horizontally or vertically).

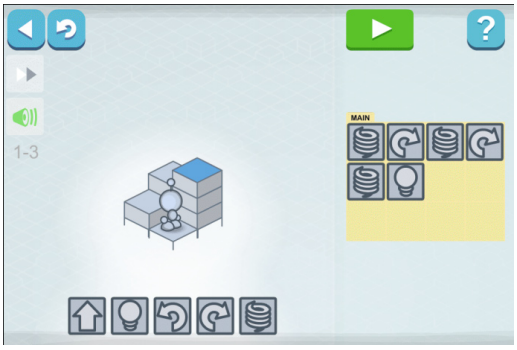

Fig. 3. In *LightBot*, the player has to find a correct sequence of actions to perform so that the robot reaches the blue cell and lighten it.

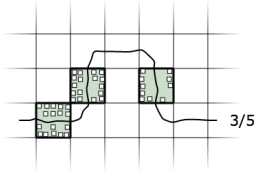

Fig. 4. In *Initial Conditions*, the player has to place cities on a map so that a river is passing through them and so that two cities cannot be adjacent. This particular map has one river and the player has to place five cities.

This problem is a typical an artificial intelligence (AI) search problem that could for example be solved using any exploration algorithms, or with a constraint programming solver, for example. Again, it is not explicitly explained to the learner who just sees a game with puzzles to solve. Playing the game will challenge the learner and should develop algorithmic thinking in learner's mind.

Again, if the purpose is educational, the provided feedback is very important. For *LightBot*, the feedback is visual since the learners can visualise the execution of their programs and therefore visually debug them if the robot failed to reach the goal. For Initial Conditions, the only feedback is a success/failed status indicating which cities are violating the constraints or which river is missing villages.

3.3. *Learn to Create Games*

Finally, another kind of online programming learning platforms offers the possibility for the users to create their own games. On these platforms, the learner has to program a game, typically with a visual programming language. For example, *Scratch* (`https://scratch.mit.edu`) uses a visual block programming language to program the behaviour of sprites.

The main window of the application where the game is shown on the left part and the code is shown on the right (Fig. 5). The behaviour of each sprite can be programmed and they can also communicate together.

Another similar platform is *Flowlab* (`http://flowlab.io`) where the game is represented by flowchart diagrams (Fig. 6). As in *Scratch*, the game is built with sprites whose individual behaviours can be programmed.

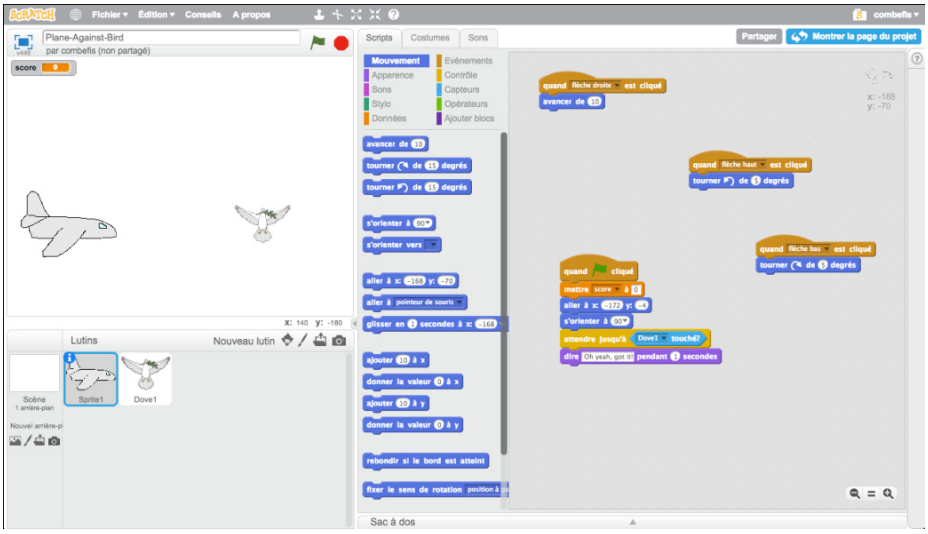

Fig. 5. The main window of the *Scratch* program is split into two parts: the left part shows the sprites of the game and the right part shows the programs representing the behaviour of the different sprites.

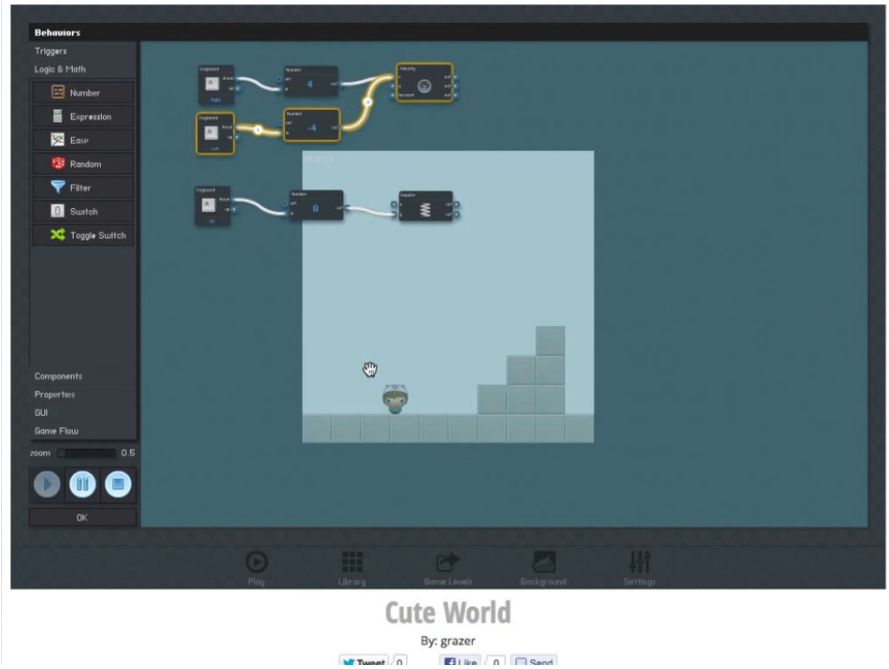

Fig. 6. In *Flowlab*, the user programs the game using flowchart diagrams representing the behaviours of the different sprites it included in the game.

This last category requires the learner to be able to program, but not necessarily to write code. The focus is put on the creativity and on the skills needed to design and architecture an application.

3.4. *Gamification of an Online Learning Platform*

Proposing an online platform is useful to make the material available to every learner. But only relying on the online aspect is not enough to motivate them to learn. As detailed in the second section, gamification can help to foster learners to make progress, increasing their engagement and motivation. Several kinds of elements can be introduced to build online game platforms:

- Points and rankings amongst players.
- Levels, grades, badges that are related to the progression.
- Live fights/contests between players or against bots.

One other important element that can be taken into account is the contextualisation of the statements of the challenges and exercises. There are indeed two main strategies that can be used to gamify an online platform to learn programming. Either game elements can be added to an already existing platform (points, rankings, badges…) or the platform can be rethought and redesigned so that to transform it completely into

a game. As highlighted by Morrison and DiSalvo (2014) who relate the gamification of the Khan Academy, one critical motivational element to have is to place the user at the centre by adding elements of pure play, to make the gaming more "playful". Examples of such approaches include *CodeSpells* (Esper *et al.*, 2013) and *Gidget* (Lee *et al.*, 2013).

Ibáñez *et al.* (2014) made an experiment to explore the impact of gamification techniques on the engagement and learning about the C programming language. The authors conclude that the gamification they performed has a positive impact on the students' engagement. Most students continued to work even after having earned the maximum amount of points. O'Donovan *et al.* (2013) present a case study where they used gamification in a university-level course about game development. In their paper, the authors provide insights about gamification mechanics and explain how the design of their game has been created and which game elements have been used. Their research allows them to conclude that the gamification they performed in a university setting was effective, increasing the engagement and understanding of the students. Again, it is a situation where the platform has been thought as a game from the starting point.

4. Online Programming Contests

This section presents several online game platforms, so that to cover the different tendencies that exist today and the characteristics presented in the previous section. The focus has been put on platforms that available for free (at least a standard version).

4.1. *Leek Wars*

Leek Wars (`http://leekwars.com/`) is a game where the player has to program the behaviour of a leek. The goal of the leek is to defeat another leek during a fight that takes place in a garden. Depending on the level of the player, his/her leek can be equipped with more weapons, health-recovering objects, etc.

The leeks are programmed in JavaScript and an API provides functions to manipulate the leak and to get information about the environment. This API is very well documented, which also trains learners to manipulate such documentation.

The platform offers the possibility to develop several AIs and to choose which one to use at any time. The player can experiment his/her AIs against bots, before jumping into the garden to fight with other real players. To motivate the player, a level system has been put in place. For each fight that is won, the player gains experience points, which allow him/her to level up. A higher level grants access to more objects such as weapons, magic spells, etc. It also improves the ranking of the player.

The game also proposes a cooperative mode where teams composed of leeks belonging to different players can fight against other teams. Such cooperative mode is another technique used in games to foster people to play regularly.

Fig. 7. Two leeks are fighting in a war, playing alternatively one after the other. The garden is divided into cells between which the leeks are moving and has some impassable obstacles.

4.2. *CodinGame*

CodinGame (`https://www.codingame.com`) proposes coding challenges to solve. Basically, the user is asked to code agents that have to interact in a given environment in order to achieve a given goal. In the example shown in Fig. 8, the user has to code the spaceship so that to destroy all the targets on the ground. The spaceship is moving from

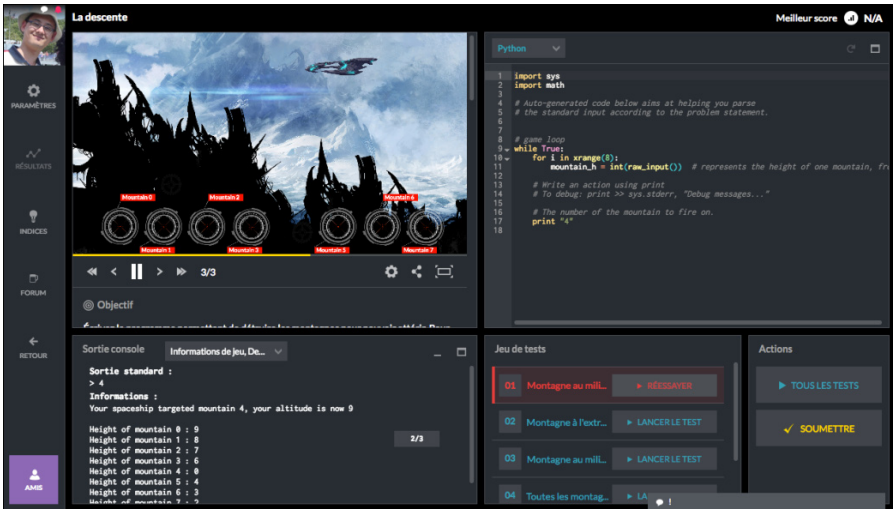

Fig. 8. In this mission, the agent is a spaceship that must destroy targets on the ground. These targets have different heights so that the spaceship must destroy them in the right order if it does not want to enter in collision with any of them.

left to right then back from right to left while descending when it reaches the rightmost part of the world. The difficulty is that targets have different heights and the spaceship must avoid entering in a collision with any of them.

Several test sets are provided and to succeed, the code of the player must pass them all (some being public and others being hidden). The execution of the code also results in an animation, which makes it possible for the learner to directly visualise the behaviour he/she wrote. This latter point helps to increase the motivation and engagement of the learner, better than a simple textual test execution report. Finally, the platform offers the possibility to code the agents with a lot of different programming languages.

Several challenges/missions are available for the user to tackle. They are organised into categories such as tutoring, advanced, etc. The users can follow their progress and that motivates to solve more challenges.

4.3. *Code Hunt*

CodeHunt (`https://www.codehunt.com`) is developed by Microsoft Research and propose coding challenges driven by tests. The goal of the game is to guess what the code must do, and then fix it until it passes all the tests. Fig. 9 shows the main view of Code Hunt where the code is on the left and the results of test execution are shown on the right. Pressing the "*Capture code*" button triggers the execution of a bunch of tests, shown on the right.

When the user solves a yet unsolved puzzle, he/she gains points so that to increase his/her position in the ranking. Moreover, some puzzles are initially locked and will only unlock when the user has solved enough puzzles. This possibility to unlock puzzles is an incentive for the user that motivates him/her to progress.

Fig. 9. The main window of Code Hunt is split into two parts: the left part shows the code being run and the right part shows the result of the execution of the public tests.

4.4. *Code Fights*

Code Fights (`https://codefights.com`) is a platform where users can participate in fights against bots or other real players. A fight typically consists in three puzzles to solve. One fight typically consists of three challenges. As shown in Fig. 10, the main window shows the statement of the challenge on the right and the code with the results of public tests on the right.

The learner can train him/herself by fighting bots or can defy any other player for a fight. There are two kinds of bots: simple ones and others developed by companies. If you manage to beat a company bot, you got a chance to be contacted by the company to possibly get a job. Another interesting feature proposed by the platform is tourney, where you compete with several other real players on the same challenges. To win the tourney you have to solve all the challenges as fast as possible.

4.5. *Discussion and Comparison*

The presented online platforms have some common points and differences. First of all, they are all in the *"learn algorithmic thinking"* category described in the previous section. Then, the level of gamification of the different platforms is quite different. A platform that has been completely designed with games in mind is one that completely agrees with the definition of gamification, as detailed in the previous section. The first one, namely *Leek Wars*, is a full online game with cooperative aspects where the ultimate goal is for the player to level up to become the best leek, the one with the most

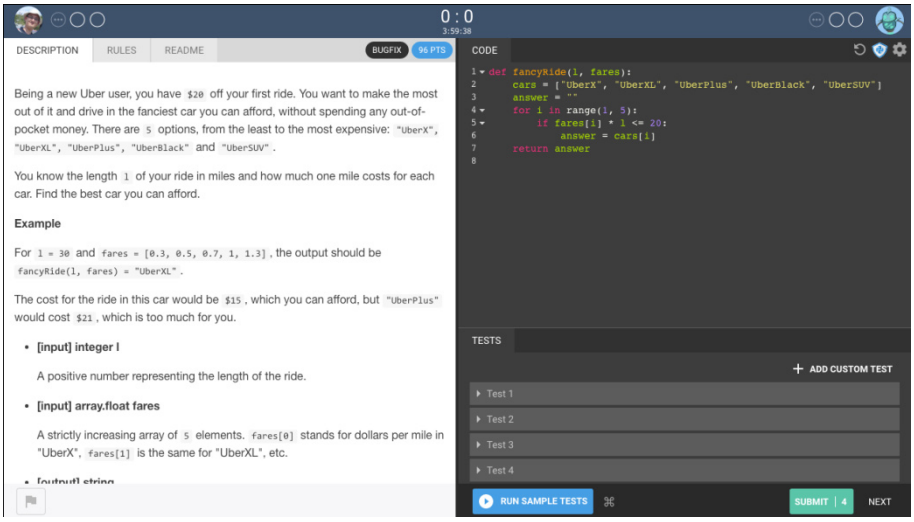

Fig. 10. The main window of *Code Fights* is split into two parts: the left part shows the statement of the challenge and the right part shows the code and the result of the execution of the public tests.

intelligent behaviours. The three other platforms do contain some game element, but they are not designed completely as full games. Table 1 summarises the game elements that appear in the presented platforms.

All the platforms presented in this paper can also be classified according to the adapted and extended schema we proposed to classify educational programming games. Table 2 shows how that classification can be done. All the games presented in this paper

Table 1

The four presented online game platforms to learn different skills of coding training and contains different game elements

Platform	Trained skills	Game elements
Leek Wars	Coding agents' AI Inventing algorithms	Points and rankings Challenges against bots and other players Cooperative mode
CodinGame	Understanding specifications Coding objects' behaviours	Points, trophies, badges, rankings Levels and achievements Matches against other players
Code Hunt	Fixing bugs Code recovery Understanding tests sets results	Points and rankings Levels with challenges to unlock
Code Fights	Understanding specifications Coding algorithms Fixing bugs Code recovery	Points, levels and rankings Matches against other players or bots

Table2

Classification of the platforms

Game	Modality	Interaction style	Environment	Learning approach
Codecademy	Visual	Keyboard/Mouse	Social Presence 2D Online	Single Player
Code Fights	Visual	Keyboard/Mouse	Social Presence 2D Online	Mixed Player
Code Hunt	Visual	Keyboard/Mouse	2D Online	Single Player
CodinGame	Visual	Keyboard/Mouse	Social Presence 2D Online	Single Player
Initial Conditions	Visual	Keyboard/Mouse	2D Online	Single Player
Leek Wars	Visual	Keyboard/Mouse	2D Online	Mixed Player Collaboration
LightBot	Visual	Keyboard/Mouse	2D Online	Single Player

are visual ones with which the players interact with the keyboard and mouse. Concerning the environment, all the games are of course online ones. They are also all 2D games and some of them will require social presence, in particular for those where duels are available. Finally, the learning approaches of those selected games can be either single or mixed player. One of the game, namely *Leek Wars*, supports collaboration through the creation of teams of Leeks.

5. Recommendations and Conclusions

To conclude this paper, the learning of programming is accessible not only to higher education students, but it is accessible to others as well. A lot of online platforms have been developed and made accessible on the Internet. In addition to the classical learning platform or MOOCs, a broad range of game-based online platforms have emerged. The main advantage of such platforms is that they foster their users to learn and keep progressing, making programming fun.

Tasks and challenges proposed by these games-based platforms could inspire tasks proposed in the IOI for two main reasons. First, contestants to the IOI are probably using those platforms so that they may be expecting similar tasks at the IOI. Second, the motivational aspect of those tasks could be transferred to the IOI contest, also the making it more motivational and interesting for its contestants.

We referred and analysed papers on gamification: contests; online platforms; online, educational, programming and serious games. These are our main findings from these papers for educational or serious programming games to be successful, motivating to learn programming and so on:

a. Feedback and assessment is very important in any educational games. Assessment may be automated or not, but it raises efforts of students. Feedback provides information about the learning process.
b. Games must have aesthetics – game must be fun to engage and improve intrinsic motivation to learn computer science or programming.
c. Collaborative educational games raise the participation of students.
d. Participation is necessary for motivation to learn as well as for reducing high dropout rates of programming students.
e. Multiplayer collaborative games are more motivating and engaging than single-player appropriate games.
f. Guidance in educational game helps players not to feel confused while they are playing.
g. Aesthetics make any educational game "*fun*". It involves players into the game.
h. Avoid negative consequences, because otherwise players may perform low.
i. Involve music in game design and the players will probably play the game longer.
j. The level of challenge neither ought to be not too low nor too high (it keeps the interest of players).
k. Games with multiple modalities, adaptive or personalized, based on real world sensory data may be more successful.

l. Contests and duels in programming games may be used to improve the programming skills.

We proposed taxonomy of the programming educational games and classified several online programming games / platforms. All of these games are visual; interactive with keyboard/mouse, 2D and online. The majority of them are designed for a single player, however, two of them are designed for mixed players (game may be played as single player or multiplayer) and one of them involves collaboration aspect. As we already mentioned, multiplayer and collaborating games are more motivating. 3D-games sometimes have more advantages than 2D, but it depends more on a combination of many other mentioned or not mentioned factors.

Our findings may be useful in creating educational programming games on your own, or recreating non-educational games and gamifying them to teach programming, it is especially useful for novices in programming. This research does not propose to use all these factors, because some of them may be successful in one kind of games, but may not be successful for other type of games. However, it is recommended to use at least some of these factors due to our research analyse.

References

ACM/IEEE Computer Society (2013). *Computer Science Curricula 2013.Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM Press, New York, N. Y. USA. Available on the internet: `https://www.acm.org/education/CS2013-final-report.pdf`

Alvarez, J., Michaud, L. (2008). Serious games: advergaming, edugaming, training, and more, *IDATE, BP 4167, 34092 Montpellier Cedex 5*. France, 3–6, 11–12.

Arena, D.A., Schwartz, D.L. (2013). Experience and explanation: using videogames to prepare students for formal instruction in statistics. *Journal of Science Education and Technology*, 1–11.

Azmi, S., Iahad,N.A., Ahmad, N. (2015). Gamification in online collaborative learning for programming courses: a literature review. *ARPN Journal of Engineering and Applied Sciences*, 10(23), 1–3.

Barata, G., Gama, S., Jorge, J., Goncalves, D. (2013). Engaging engineering students with gamification. In: *Proceedings of the 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES 2013)*. 1–8.

Becker, K. (2008). Video game pedagogy: good games = good pedagogy. In: T.C. Miller (Ed.), *Dames: Purpose and Potential in Education*. Springer, NewYork, NY, 73–125.

Bishop, J., Horspool, R., Xie, T., Tillmann, N., de Halleux, J. (2015). Code hunt: experience with coding contests at scale. In: *Proceedings of the 37th International Conference on Software Engineering (ICSE 2015)*. 398–407.

Combéfis, S., Bibal, A., Van Roy, P. (2014). Recasting a traditional course into a MOOC by means of a SPOC. In: *Proceedings of the European MOOCs Stakeholders Summit 2014 (EMOOCs 2014)*. 205–208.

Combéfis, S., Saint-Marcq, D. (2012). Teaching programming and algorithm design with Pythia, a web-based learning platform. *Olympiads in Informatics*, 6, 31–43.

Combéfis, S., Van den Schrieck, V., Nootens, A. (2013). Growing algorithmic thinking through interactive problems to encourage learning programming. *Olympiads in Informatics*, 7, 3–13.

Combéfis, S., Wautelet, J. (2014). Programming trainings and informatics teaching through online contest. *Olympiads in Informatics*, 8, 21–34.

Cuba-Ricardo, G., Serrano-Rodríguez, M.T.,Leyva-Figueredo, P.A., Medoza-Tauler, L.L. (2015). Methodology for characterization of cognitive activities when solving programming problems of an algorithmic nature. *Olympiads in Informatics*, 9, 27–30.

Deterding, S., Dixon, D., Khaled, R., Nacke, L. (2011). From game design elements to gamefulness: Defining "gamification". MindTrek, 2.

Dickey, M.D. (2005). Engaging by design: how engagement strategies in popular computer and video games can inform instructional design. *Educational Technology Research and Development*, 53(2), 67–83.

O'Donovan, S., Gain, J., Marais, P. (2013). A case study in the gamification of a university-level games development course. In: *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference (SAICSIT 2013)*. 242–251.

Esper, S., Foster, S., Griswold, W. (2013). CodeSpells: embodying the metaphor of wizardry for programming. In: *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2013)*. 249–254.

Gee, J.P. (2003). What video games have to teach us about learning and literacy. *Computers in Entertainment* (CIE), 1(1), 20.

Gee, J.P. (2005). Good video games and good learning. *Phi Kappa Phi Forum,* 85(2), 33–37.

Hunicke, R., Leblanc, M. Zubek, R. (2004). MDA: A formal approach to game design and game research. In *Proc. AAAI workshop on Challenges in Game*. AAAI Press, 1–5.

Ibáñez, M.-B., Di-Serio, Á., Delgado-Kloos, C. (2014). Gamification for engaging computer science students in learning activities: a case study. *IEEE Transactions on Learning Technologies*, 7(3), 291–301.

Laamarti, F., Eid, M., El Saddik, A. (2014). An overview of serious games. *International Journal of Computer Games Technology*. Article ID 358152, vol. 2014, 1–15. Available on the internet: `http://dx.doi.org/10.1155/2014/358152`

Lee, M., Ko, A., Kwan, I. (2013). In-game assessments increase novice programmers' engagement and level completion speed. In: *Proceedings of the 9th Annual International ACM Conference on International Computing Education Research (ICER 2013)*. 153–160.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., Resnick, M. (2004). Scratch: a sneak review. In: *Proceedings of the 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives (ISSEP 2011)*. 155–164.

Mora, A., Riera, D., González, C, Arnedo-Moreno, J. (2015). A literature review of gamification design frameworks. In: *7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games)*, *At Skövde*. IEEE, 1–8.

Morrison, B., DiSalvo, B. (2014). Khan Academy Gamifies Computer Science. In: *Proceedings of the 45th Technical Symposium on Computer Science Education (SIGCSE 2014)*. 39–44.

Muratet, M., Torguet, P., Viallet, F., Jessel, J.-P. (2010). Experimental feedback on Prog&Play: a serious game for programming practice. In: L. Kjelldahl and G. Baronosk (Eds.), *EUROGRAPHICS.* 1–8.

Nah, F., Zeng, Q., Telaprolu, V., Ayyappa, A., Eschenbrenner, B. (2014). Gamification of education: a review of literature. In: *Proceedings of the First International Conference on HCI in Business (HCIB 2014)*. 401–409.

Prensky, M. (2003). Digital game-based learning. *Computers in Entertainment* (CIE), 1(1), 21.

Ravaja, N., Saari, T., Laarni, J., Kallinen, K., Salminen, M., Holopainen, J., Järvinen, A. (2005). The psychophysiology of video gaming: phasic emotional responses to game events. In: *Proceedings of DiGRA 2005 Conference: Changing Views – Worlds in Play*. Vancouver, Canada, 3, 1–13.

Sawyer, B., Rejeski, D. (2002). *Serious Games: Improving Public Policy through Game-Based Learning and Simulation*. Woodrow Wilson International Center for Scholars, Washington, DC, USA.

Schell, J. (2008). *The art of Game Design: A Book of Lenses*. Carnegie Mellon University and Schell Games, Pittsburgh, Pennsylvania, USA.

Shute, V.J. (2011). Stealth assessment in computer-based games to support learning. In: S. Tobias and D. Fletcher (Eds.).*Computer Games and Instruction*. Information Age: Charlotte N. C., 503–524.

Squire, K. (2008). Open-ended video games: a model for developing learning for the interactive age. In: K. Salen (Ed.), *The John D. and Catherine T. MacArthur Foundation Series on Digital Media and Learning*. MA: The MIT Press, Cambridge, 167–198.

Surakka, S., Malmi, L. (2004). Cognitive skills of experienced software developer: Delphi study. In: Korhonen A., Malmi, L. (Eds), *Kolin Kolistelut–Koli Calling 2004*. In: *Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education*. Finland, Koli, 37–46.

Swacha, J., Baszuro, P. (2013). Gamification-based e-learning platform for computer programming education. In: *Proceedings of the X World Conference on Computers in Education (WCCE 2015)*. 122–130.

Tillmann, N., de Halleux, J., Xie, T., Bishop, J. (2014). Code hunt: gamifying teaching and learning of computer science at scale. In: *Proceedings of the First ACM Conference on Learning @ Scale Conference (L@S 2014)*. 221–222.

Tillmann, N., De Halleux, J., Xie, T., SumitGulwani, Bishop, J. (2013). Teaching and learning programming and software engineering via interactive gaming. In: *35th International Conference on Software Engineering (ICSE)*. IEEE, San Francisco, CA, 1117–1126.

Vihavainen, A., Airaksinen, J., Watson, Ch. (2014). A systematic review of approaches for teaching introductory programming and their influence on success. In: *ICER '14Proceedings of the tenth annual conference on International computing education research*. ACM, New York, NY, 19–26.

S. Combéfis obtained his PhD in engineering in November 2013 from the Université catholique de Louvain in Belgium. He is currently working as a lecturer at the École Centrale des Arts et Métiers (ECAM), where he is mainly teaching informatics. He also got an advanced master in pedagogy in higher education in June 2014. He founded the Belgian Olympiad in Informatics (be-OI) with Damien Leroy in 2010. In 2012, he introduced the Bebras contest in Belgium and at the same time he founded the CSITEd non-profit organisation that aims at promoting computer science in secondary schools.

G. Beresnevičius is a doctoral student of informatics engineering at the Institute of Mathematics and Informatics of Vilnius University. He obtained his master degree in Mathematics and Informatics Didactics with Cum Laude diploma in Vilnius University. His research focuses on teaching informatics, especially programming through games, as well as the theories of serious games and gamification. He participated in several international conferences and presented his research. He likes programming websites using HTML, CSS, JavaScript, PHP and MySQL languages.

V. Dagienė is head of the Informatics Methodology Department at the Institute of Mathematics and Informatics of Vilnius University. Her current research is in Computer Science Education, focusing on cognitive aspects of algorithmic thinking as well as computational thinking. She has published over 200 scientific papers and methodological works, has written more than 50 textbooks in the field of informatics and information technology for primary and secondary education. She works with various expert groups all over the world. In 2004, she established the Bebras Challenge on Informatics and Computational Thinking, which is successfully implemented in more than 50 countries. She has participated in several EU-funded R&D projects, as well as in a number of national research studies connected with technology and education.