ACTA DIDACTICA NAPOCENSIA

Volume 13, Number 1, 2020 - DOI: 10.24193/adn.13.1.7

THE EFFECTIVENESS OF A FULLY GAMIFIED PROGRAMMING COURSE AFTER COMBINING WITH SERIOUS GAMES

Sándor KIRÁLY and Tamás BALLA

Abstract: Our online programming language courses have been developed for youngsters who are interested in computer programming. The courses were gamified with some common gamification elements: points, badges, incentives, immediate feedback and leaderboard. The developed Learning Management System (LMS) for our learning portal provides the chance to monitor all users' activities. After examining our registered users' progress in the courses, we tried to increase the effectiveness of the courses (C+, C# and Java) with developing educational game programs that cannot only foster motivation during the playing process but provide faster understanding, since students play with what they need to learn and understand. We compared the performance of the last 200 students who registered and completed the last chapter in the curriculum before the game programs had been added into the Java course, and the first 200 students, who registered and completed the last chapter in the curriculum, after the addition of the games in the supported coding tasks. The results were impressive and proved that with our thematic didactic game programs, users were able to solve coding tasks more effectively.

Key words: gamification, educational games, computer programming, serious games

1. Introduction

One of the most difficult challenges faced by most students is the understanding of programming fundamentals, particularly for novices (Zapušek and Rugelj, 2013; Coelho et al., 2011; López et al., 2018). They find it hard to learn programming languages since concepts are complex and cognitively demanding, and are totally different from what they are accustomed to. Therefore, introductory courses are usually considered difficult and often have high dropout rates. Learning computer programming requires algorithmic thinking, problem-solving skills, and it is a long-term process (Muratet et al., 2011). There is a distinction between programming knowledge (for example being able to state how a "while" loop works) and programming strategies (for example using a "while" loop in a program). Students may have difficulties with combining the programming constructs (conditionals, loops, etc.) into viable solutions. Winslow (Winslow, 1996) observed that students who understand the syntax and semantics of individual programming statements sometimes cannot translate it into software code.

In an online environment, students' achievement can be influenced by numerous factors that stem from their individual personality. These factors can be for example their ability to maintain their attention or their intrinsic motivation. One of the most important factors that should be considered is the students' engagement: this can be defined as the student's cognitive process, active participation and emotional involvement in the learning process (Pellas, 2011). In an online environment, it is also important to increase the student's cognitive involvement (Wolf, 2011) that can be carried out, for example, by applying gamified elements. The popularity of a computer programming portal stems from its effectiveness that can be improved applying gamification according to researchers.

The term gamification was coined by Nick Pelling in 2002 (Pelling, 2011). Gamification refers to the use of game elements in a non-game context in order to increase engagement between human and computer (Deterding et al., 2011) and to motivate students in an educational setting (Tsai et al., 2016.). Gamification uses elements that support intrinsic and extrinsic motivation, for example, offering prizes (rewards) (Surendeleg et al., 2014). It also offers an opportunity to experiment with rules, emotions and social roles (Lee et al., 2011). To summarise the different definitions we can say that gamification

Received February 2020.

Cite as: Kiraly, S; & Balla, T. (2020). The effectiveness of a fully gamified programming course after combining with serious games. *Acta Didactica Napocensia*, 13(1), 65-76, https://doi.org/10.24193/adn.13.1.7

is the use of elements of game design in non-game context which differentiates it from games and the design for playful interactions.

Due to the above-mentioned advantageous features of gamification, we have made our programming courses gamified. They contain game elements in a non-game context: points, badges, incentive and immediate feedback and leaderboard. Besides, after completing the different sections, users get information and screenshots about the downloadable, reward program that would become available for students after completing the course.

There are numerous games types that can enhance the effectiveness of a programming course. Games that use some kind of computing machinery (e.g., personal computer, a smartphone or a piece of electronics dedicated to playing games such as a video game console) are called **digital games**. According to a more modern definition: a **serious game** is a digital game created with the intention of entertaining and achieving at least one additional goal, for example learning or health (Dörner et al., 2016). These games can also be used to stimulate programming learning (Frankovic et al., 2018). To improve the effectiveness of our fully gamified curriculum, 19 games were developed for educational purposes that can be considered serious games according the aforementioned definition.

Following this definition, a serious game needs to be a digital game and its additional goals, however, do not have to be in an educational context. Other areas are also covered by this definition. Thus, they have already been developed for different fields, i.e., healthcare (Brown et al., 1997) or climate change (Wu and Lee, 2015). However, serious games are often intended for learning. For example, Disney's Minnie explores the land of Dizz (The Walt Disney Company LTD. 2014) is an example of a serious game where small children can develop problem solving skills. Some platforms, such as CodeCombat or Prog&Play may help even to recruit students in computer science (Muratet et al., 2010). The first one is a game-based computer science program where students type real code and see their characters react in real time.

According to the aforementioned definitions, our developed game programs are serious games with a developed didactic purpose. Unlike serious games, educational or didactic games are definitely educating tools serving a didactic purpose. They can be defined as interactive, competitive lessons with defined learning outcomes that enable students to have fun during knowledge acquisition. Their goal is not merely fun but they also contain an educational component (Blumberg et al., 2013).

Playing with some of our programs, users can simulate the working of computer programming code structures following the given instructions that help them to understand the working of codes. Following the definition (Sauvé, Renaud, and Kaufman (2007)), these games are not only education games but also **Instructional Simulation Games** (see Figure 1). Learners using buttons, textboxes and slide bars can run or stop the experiment, which is a simplified version of reality, and change the parameters of the phenomena within a framework of rules.

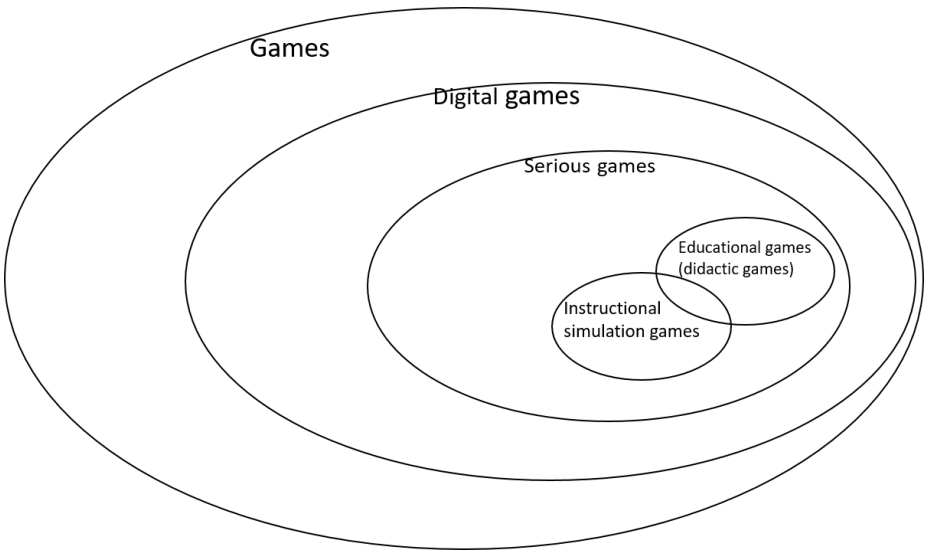

Figure 1. *Game types*

Game-based Learning (GBL) is a process and practice of learning using games. Its purpose is to improve learning, to increase learning effectiveness. Our portal cannot be considered neither only Game-based Learning (GBL) nor only a gamified one. We have extended our fully gamified curriculum with serious games that can be considered educational games and instructional simulation games.

Numerous online platforms exist for teaching programing. Some of them were developed for learning to code, others for learning algorithmic thinking or for learning to create games (Combéfis et al. 2016). The goal of the platforms that belong to the first category is to allow their users learning and to learn to code. Codecademy is an example of an online gamified platform but without serious or educational games.

2. The gamified portal for teaching programming languages

In our programming portal, students can learn how to code fundamental algorithms in a chosen language that can be C#, C++ or Java. The main objective of this site is to teach students the basic programming structures that are necessary to be able to pass the advanced ICT (*Information and communications technology*) level exam. After completing the website course, they will be able to use an IDE (*Integrated Development Environment*) and solve programming tasks. The programming knowledge gained from this portal can be enough for students to understand programming resources on the internet so they can achieve more in this field.

Our portal is available for anyone after completing the registration that requires the following data: login name, first name and family name, date of birth, gender and a password. After logging in to the portal, students need to choose the language they wish to learn in. The material to be learned can be found on the left side of the screen. The exercise belonging to the current topic is on the right side. Students are asked to write the correct code in the panel on the right side. After completing the code, they can submit the solution by clicking on the Send button. If the code is correct, the next exercise or unit will be displayed. If the code is incorrect, a message stating the type of the error will be displayed. By clicking on the Hint button, the portal gives instructions allowing for students to correct the mistake. Naturally, there is more than one exercise for each topic and these exercises are applicable to real life so as to allow students to feel the usefulness of the portal (Joo et al. 2013). Practical exercises, such as finding the closest defibrillator to save a life, writing a code that can control the descent of a spacecraft onto a planet, how to move an object from a given (X,Y) coordinate to another position as fast as possible or controlling the parking sensor of a car can increase the efficiency of the e-learning environment. For difficult tasks, we created videos and animations to motivate students to come up with a solution.

In this portal, games accompany the entire learning process. Students who complete each unit and task of the curriculum can download a game program for both Android and Windows platforms and they can trace their “downloading process” after getting new percentages as special points after completing the different sections of the curriculum. After completing a section, the portal shows different screenshot parts of the reward game to allow the users to know more about the downloadable game.

3. The developed educational games

According to Vihavainen et al. (2014), the intervention of educational games in programming courses raises pass rates by 10.8% on average. We were curious, whether developing and adding educational games to our portal can improve its effectiveness? Therefore, we developed 19 educational games for each unit and the three programming languages. They were written in HTML5 and JavaScript and added to the course immediately before a coding task. In this paper, we introduce five of them with their learning objectives and learning skills.

3.1. Let the ball roll into the computer

Learning goal: read values from the keyboard using different I/O (Input/Output) statements.

Activity: the value has to be read from the keyboard according to the game request, the player has to choose the appropriate method and the returned value type (see Figure 2).

The name of the game is *Input* in Table 2.

This game has been inserted to the Java course immediately before the following exercises.

Coding task 1: Declare three integer variables with 0 initial value that store the value of the sides of a triangle. The names of the variables are: *length_a*, *length_b*, *length_c*. Your program should read three numbers from the standard input into the given variables.

Coding task 2: Declare a variable with the name *nice* where we need to use a text that is read from the standard input. In the second line, write the code that reads the user-defined text from the standard input.

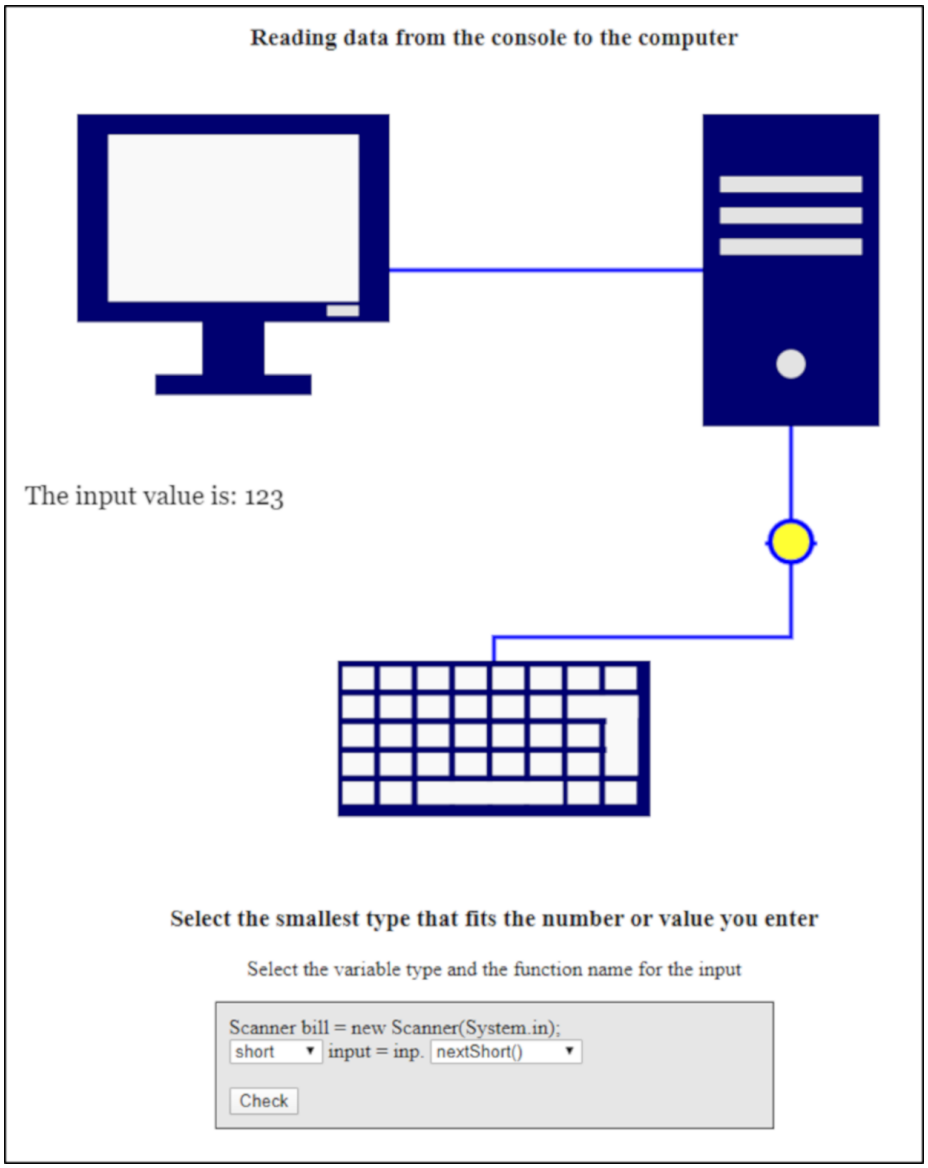

Figure 2. *Let the ball roll into the computer by finding the appropriate statement*

3.2. Memory allocation

Learning goal: how to declare different types of variables and understanding what happens after running the declaration and value assignment statements.

Activity: player has to enter the proper declaration or assignment statement (see Figure 3).

The name of the game is *Memory* in Table 2. This game has been inserted to the Java course immediately before the following exercises.

Coding task 1: We want to store a Boolean value for each spacecraft, depending on whether they are within a certain distance. The variable name is near. The initial value of the variable must be false. Enter the statement.

Coding task 2: A program asks you the name of your mobile service provider. We need to store the company name in a string type. The start value of the variable must be empty. The name of the variable should be *company*. Enter the statement.

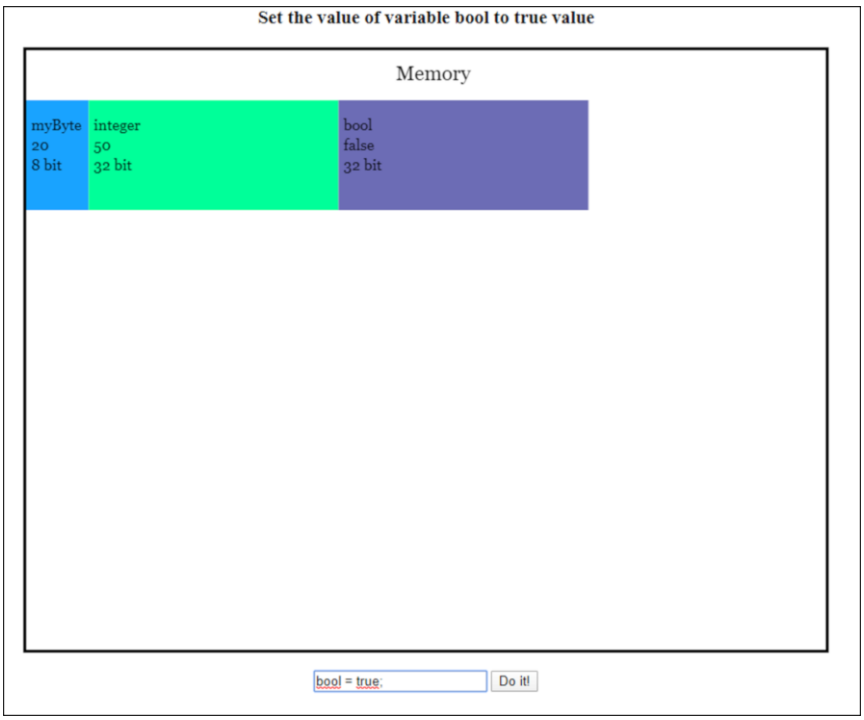

Figure 3. *Memory allocation game*

3.3. Writing and calling a procedure with two parameters

Learning goal: how to call a procedure with two parameters, and what happens after calling a method. In the game, the yellow ball represents the line of the programming statement to be currently executed.

Activity: player has to set the value of the current parameter, the type of the formal parameter and the displayed text (see Figure 4).

The name of the game is *Method2* in Table 2. This game has been inserted to the Java course immediately before the following exercises.

Coding task 1: We know where defibrillators are located in a city. To be precise, in the array defib (its elements are integer type), we store how far they are from a given point. Write a function (with name *Find_min*) that takes the array *defib* of type integer as a parameter and returns the distance of the nearest defibrillator, that is a long value and will be returned by this function.

Coding task 2: In the shooter task, the sign + meant that the person had hit the target, the sign – meant that he did not. For example, string "---++++" means, that the first three shots failed unlike the last three shots. Your task is to write a function that takes the string and a character that is either '+' or '-' and returns true or false depending on whether the passed character occurs in the string. All you have to do now is to write the header of the function. The name of the function is *counter_p*, the names of its (formal) parameters are: *results* and *sign*, the type of return value is bool. Enter the single line code into the code window.

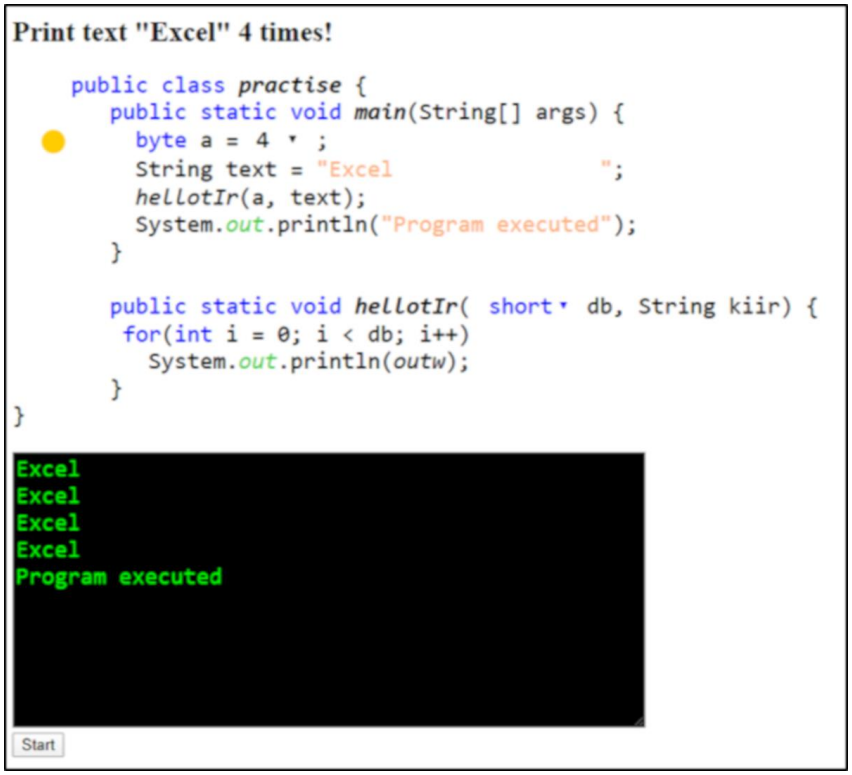

Figure 4. *Writing a procedure with two parameters*

3.4. For loop

Learning goal: how to write a for loop especially the initial value, the condition and the increment, and understanding the effect of for loop. While running the code, the current statement is denoted by blue colour.

Activity: player has to set the values belonging to a for loop according to the instruction of the exercise (see Figure 5).

The name of the game is *Forloop* in Table 2. This game has been inserted to the Java course immediately before the following exercises.

Coding task 1: There are two variables, *speed* and *altitude*, both are integers, and you do not have to declare them. Write a for loop in which you use a previously declared variable *i*, with an initial value of 1, and in the condition you need to write: *speed> 1 && altitude! = 0*. You must decrement the values of both variables by one in the header of the loop. Inside the loop, your code should display the values of both variables into the following format: The speed is 10 and the height is 2. (If the value of variable speed is 10 and the height is 2.)

Coding task 2: The aircraft ascends until its speed exceeds 1000 or its altitude reaches 2000 units. Meanwhile, the speed increases by 2 units, the height by 4 units. Their initial values are unknown. Write a program that tells us how many steps our plane will ascend. The values of *speed* and *altitude* variables are integer but unknown and have already been declared. In line 1, declare a variable with name *count* and integer type and 0 initial value. In line 2, write the for loop header. The start value of the variable *count* should be 1, the count value should be incremented by one in the header of the loop. The condition in the for loop is *speed<1000 && altitude<2001*. Inside the loop, display the value of variable count.

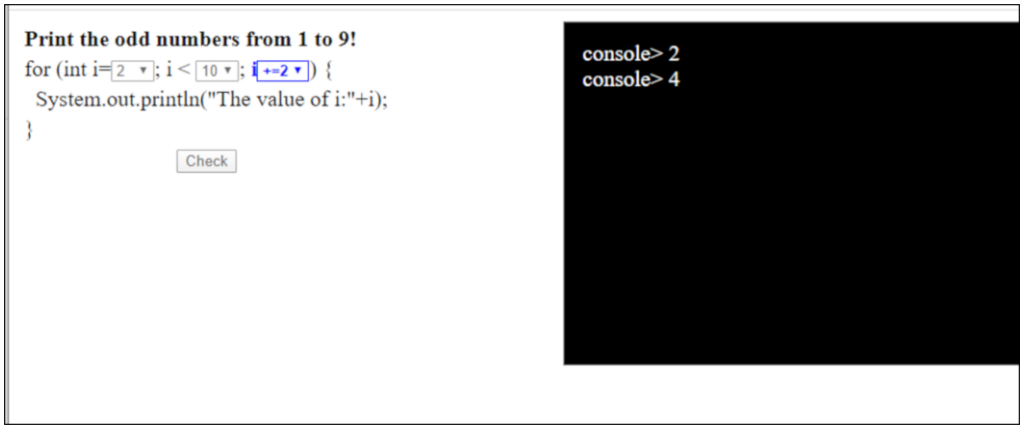

Figure 5. *For loop implementation*

3.5. Buckets

Learning goal: understanding logical operator (&& and ||) and if statement

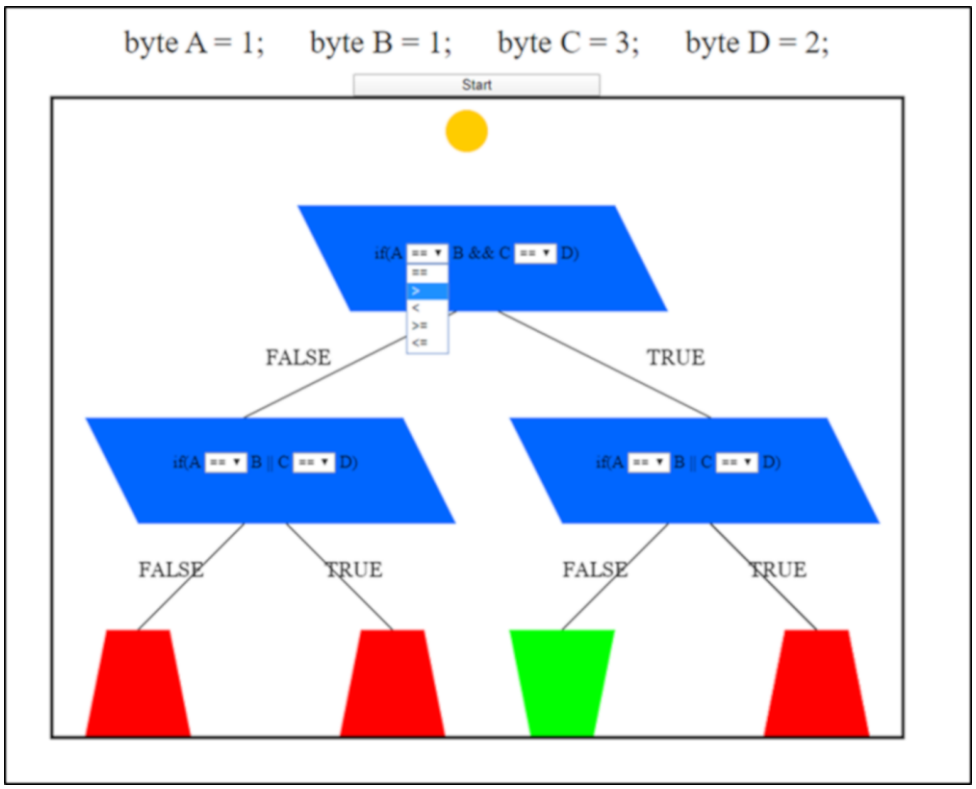

Figure 6. *Drop the ball into the green bucket*

Activity: player has to set the correct relational operators in if functions to get the ball into the green bucket (see Figure 6).

The name of the game is *If* in Table 2. This game has been inserted to the Java course immediately before the following exercises.

Coding task 1: In a car, the operation of the parking radar is sensed by sensors. The sensors indicate how far the object is from the car. In our example, the sensors do not indicate anything beyond 100 cm. If an object is within 100 cm but not closer than 50 cm, a yellow colour appears on the car display. If the object is within 50 cm (or so) but 30 cm away, a red colour appears on the screen, which will

also blink. If the object is at least 30 cm from the car, a continuous red is displayed. Write a program that declares a variable named *distance* in the first line with an initial value of 0. In the second line, read a distance value from the standard input. Starting from the next line, use statements *if* and display characters of " ", "S", "VP"or "P" considering the given intervals. If the distance is greater than 100, display " ", in the case of interval]50;100] display "S", in the case of interval]30;50] display "VP" otherwise display "P".

Coding task 2: Write a code that displays text "Let's continue" or "The End" according to the user's input. In the first line, display the text "Continue (Y/N)?". In the second line, declare a *char* type variable with name *answer,* and store the value read from the standard input. If the read value is "Y", your code should display "Let's continue" or else "The End".

4. Measuring the effect of the developed games

We have also implemented traditional board games such as Match Pairs Memory Game, Hangman and two own-developed board games, but because they were placed in the portal more than once with different contents, we could not measure their efficiency. Fifteen educational games were placed immediately before an exercise which are followed by another exercise on the current topic for which the game was created, everything else remained unchanged.

Our LMS (*Learning Management System*) is capable of the administration, documentation, tracking, reporting, and delivery of the curriculum and, since it can log the users' activities we can determine after how many attempts a user was able to solve a coding task. To measure the effectiveness of the games, we analysed the first 200 students' activities who had completed the last chapter **after** we placed the games in the Java programming course of the portal. They belong to the **experimental group**. We compared two groups of 200 students; those who played games before completing the last chapter and those did not. They belong to the **control group**. Students in both groups signed in the course freely. Table 1 shows the available data about the groups.

Table 1. *Known data for each group*

Control group		Experimental group	
Average age	**Gender – male (%)**	**Average age**	**Gender – male (%)**
21.3	74.4	21.9	73.9

Table 2 shows how many students were able to solve the first and the second exercises after the first or second try from the 200, before adding the games to the portal (control group) and after adding them to it (experimental group). The discussed games have been highlighted.

Table 2. *How many students from 200 solved coding tasks after the first or second try*

	Exercise 1				**Exercise 2**			
	Before		**After**		**Before**		**After**	
	1st attempt	**2nd attempt**	**1st attempt**	**2nd attempt**	**1st attempt**	**2nd attempt**	**1st attempt**	**2nd attempt**
Identifiers	112	113	133	139	121	144	155	161
Types	134	169	143	181	141	171	148	181
Memory	123	145	145	161	119	141	145	155
Declare	124	155	131	159	121	155	129	159
Input	97	131	123	151	103	141	128	166
Operators	130	134	131	152	134	155	155	161
If	119	145	138	167	132	149	145	161

Forloop	131	151	143	166	136	149	140	155
Whileloop	123	149	139	162	131	166	147	170
Method1	99	135	111	155	102	151	119	160
Method2	121	155	135	173	131	139	145	156
Method3	120	145	131	156	117	134	131	146
Method4	131	141	141	144	141	149	145	157
Exception	89	121	90	122	94	131	111	133
Files	134	156	137	167	121	154	122	159
Average	**119**	**143**	**131**	**157**	**123**	**149**	**138**	**159**

For example, the first coding task we added after the Identifers game was solved at the first try by 112 students without using the game and 113 students at the first or the second try (another 1 student). (The remaining 87 students attempted more.) From 200 students, 133 (versus 112) were able to solve this coding task at the first try and 139 at the first or the second try after playing the game. (The remaining 61 students attempted more.) The data show an impressive improvement not only in the case of the presented games.

Table 3 shows the same results in percent.

Table 3. *What percentage of the students from 200 solved coding tasks after the first and the second try*

	Exercise 1				**Exercise 2**			
	Before		**After**		**Before**		**After**	
	1st attempt	**2nd attempt**	**1st attempt**	**2nd attempt**	**1st attempt**	**2nd attempt**	**1st attempt**	**2nd attempt**
Identifiers	56%	57%	67%	70%	61%	72%	78%	81%
Types	67%	85%	72%	91%	71%	86%	74%	91%
Memory	62%	73%	73%	81%	60%	71%	73%	78%
Declare	62%	78%	66%	80%	61%	78%	65%	80%
Input	49%	66%	62%	76%	52%	71%	64%	83%
Operators	65%	67%	66%	76%	67%	78%	78%	81%
If	60%	73%	69%	84%	66%	75%	73%	81%
Forloop	66%	76%	72%	83%	68%	75%	70%	78%
Whileloop	62%	75%	70%	81%	66%	83%	74%	85%
Method1	50%	68%	56%	78%	51%	76%	60%	80%
Method2	61%	78%	68%	87%	66%	70%	73%	78%
Method3	60%	73%	66%	78%	59%	67%	66%	73%
Method4	66%	71%	71%	72%	71%	75%	73%	79%
Exception	45%	61%	45%	61%	47%	66%	56%	67%
Files	67%	78%	69%	84%	61%	77%	61%	80%
Average	**60%**	**72%**	**66%**	**79%**	**61%**	**74%**	**69%**	**79%**

Examining the table, it can be seen that all games have increased efficiency, for example 68% of students were able to solve the coding task at the first try after playing the aforementioned Method2

game and only 61% of students without playing this game, or for example, 73% of students were able to solve the first coding task at the first try after playing the aforementioned Memory game and only 62% of students without playing this game.

Let *E1B1* denote the percentage of students who completed Exercise 1 at the first attempt before playing games and let *E1A1* denote the same but for those who played games. Also, let *E1B2* be the percentage of students who completed exercise one after two attempts before playing the game and *E1A2* is the same for the group that played games. Similarly, *E2B1*, *E2A1*, *E2B2*, *E2A2* are variables for exercise two. To statistically test the difference in the mean percentage of students who completed the exercises before and after playing the games, paired **t-test** is used. There are four hypotheses to test:

H0: The mean percentage of students who completed Exercise X after Y attempt(s) without playing the games is the same as the mean percentage of those who played the game.

H1: The mean percentage is different.

Where X and Y are either one or two.

The observations are assumed to be independent. Since the sample size is small (15), the assumption of normality has to be tested to validate the results of the paired t-test. Shapiro-Wilk test was carried out to test if the difference between the paired values is normally distributed or not, i.e. the normality of *E1B1-E1A1*, *E1B2-E2A2*, *E2B1-E2A1*, *E2B2-E2A2*. The results suggest that all the differences follow a normal distribution.

The results of the paired t-test are shown in Table 4 below:

Table 4. *The results of paired t-test*

Pair	Mean difference	Test statistics	p-value
E1B1-E1A1	0.063	-6.504	<0.001
E1B2-E1A2	0.069	-7.113	<0.001
E2B1-E2A1	0.074	-6.114	<0.001
E2B2-E2A2	0.05	-6.404	<0.001

The p-value of all pairs is less than 0.001, so *H0* can be rejected at a 5% significance level in favour of *H1*. This means that the average number of students who completed the exercise after playing the game was higher than the average number of those who completed the exercise without playing. This applies to both exercise one and two.

5. Conclusion

The aim of this study is to demonstrate that the effectiveness of a gamified programming course can be enhanced with educational games. Combining educational games with gamification can improve the effectiveness of a gamified learning portal and raise gamification to a new, higher level.

Educational games stimulate active learning and presentation of learning content in different contexts, which are interesting and engaging for students. We can use the game format to teach computer programming in a way which implements serious games or “simple” instructional simulation games to help students in algorithmic thinking and problem solving. Our developed games actively promote interactivity and deeper learning.

This paper shows that in a gamified programming online course, after adding didactic games to the portal, more students were able to complete the coding tasks at first and second try than was previously the case. Performing a paired t-test, in each case, there was a strong evidence that the effectiveness of the otherwise fully gamified curriculum was further improved as a result of our educational games.

References

Blumberg, F., Debby E. Almonte, Jared, S. Anthony, and Hashimoto, N. (2013). Serious Games: What Are They? What Do They Do? Why Should WE Play Them?. in: *Dill, K.E. (Hrsg.): The Oxford Handbook of Media Psychology,* Oxford et al. 2013. S. 334-351. doi: 10.1093/oxfordhb/9780195398809.013.0019

Brown, S.J., Lieberman, D.A., Gemeny, B.A., Fan, Y.C., Wilson, D.M. & Pasta, D.J. (1997). Educational Video Game for Juvenile Diabetes: Results of a Controlled Trial. *Med. Inform. 1997, 22, 77–89.*

Coelho, A., Kato, E., Xavier, J. & Gonçalves, R. (2011). Serious Game for Introductory Programming. *6944. 61-71*. doi: 10.1007/978-3-642-23834-5_6

Combéfis, S., Beresnevicius, G, & Dagniene, V. (2016). Learning Programming through Games and Contests: Overview, Characterisation and Discussion. *Olympiads in Informatics, 2016,* Vol. 10, 39–60. doi: 10.15388/ioi.2016.03

Deterding, S., Khaled, R., Nacke, L., Dixon, D. (2011). Gamification: Toward a definition. in: *Proceedings of Chi 2011,* Workshop Gamification: Using Game Design Elements in Non-Game Contexts. 6-9.

Dörner, R., Göbel, S., Effelsberg, W., Wiemeyer, J. (2016). SeriousGames—Foundations,Concepts and Practice. *1st ed.; Springer International Publishing: Basel, Switzerland*. doi:10.1007/978-3-319-40612-1

Frankovic, I., Hoic-Bozic, N., Nacinovic Prskalo, L. (2018). Serious Games for Learning Programming Concepts. *International Conference The Future of Education 8th Edition Florence,* Italy 28-29 June 2018 ISBN: 978-88-3359-020-2 ISSN: 2384-9509.

Joo, Y.J., JOUNG, S. & KIM, E.K. (2013). Structural Relationships among E-learners' sense of Presence, Usage, Flow, Satisfaction, and Persistence. *Educational Technology and Society,* 16(2), 310-324., 2013.

Lee, J., & Hammer, J. (2011). Gamification in education: what, how, why bother?. *Acad. Exch. Q. 15* (2), 1–5.

López, A., Rincón, F. & Elvira G. (2018). Gamification as Learning Scenario in Programming Course of Higher Education. *LNCS, volume 10925,* doi: 10.1007/978-3-319-91152-6_16

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.

Muratet, M., Torguet, P., Viallet, F. & Jessel, J-P (2011). Experimental Feedback on Prog&Play: A Serious Game for Programming Practice. *Comput. Graph. Forum. 30.* 61-73. doi: 10.1111/j.1467-8659.2010.01829.x

Pellas, N. (2011). The influence of computer self-efficacy, metacognitive self-regulation and self-esteem on student engagement in online learning programs. *Evidence from the virtual world of Second Life Computers in Human Behaviour,* 35, 157-170, 2014, doi: 10.1016/j.chb.2014.02.048

Pelling, N. (2011). The (short) prehistory of "gamification"', Funding Startups (& other impossibilities). Available at: https://nanodome.wordpress.com/2011/08/09/the-short-prehistory-of-gamification/ [2020.04.13.]

Sauvé, L., Renaud, L., Kaufman, D., & Marquis, J. S. (2007). Distinguishing between games and simulations: A systematic review. *Educational Technology & Society*, 10 (3), 247256.

Surendeleg, G., Murwa, V., Yun & H.K., Kim, Y.S. (2014). The role of gamification in education–a literature review. *Contemporary. Engineering Sciences* Vol 7 No 29–32, pp 1609-1616.

Tsai, M.-J., Huang, L.-J., Hou, H.-T., Hsu, C.-Y., Chiou, G.-L. (2016). Visual behavior, flow and achievement in game-based learning. *Computers & Education. 98*, pp 115–129. doi: 10.1080/00461520.2015.1122533

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.

Winslow, L.E. (1996). Programming pedagogy – A psychological overview. *SIGCSE Bulletin*, Vol. 28, pp 1722.

Wolf, M. (2007). Learning to Think in a Digital World. in: *Bauerlein, M. (ed.), 'The digital divide: arguments for and against Facebook, Google, texting, and the ages of social network'* Jeremy P. Tarcher/Penguin, New York. 34-37., 2007.

Wu, J.S.; Lee, J.J. (2015). Climate Change Games as Tools for Education and Engagement. *Nat. Clim. Chang.*, 5, 413–418.

Zapušek, M. & Rugelj, J. (2013). Learning programming with serious games. *Transactions on Game Based Learning.* doi: 10.4108/trans.gbl.01-06.2013.e6

Authors

Sándor KIRÁLY, Eszterházy Károly University, Faculty of Informatics, Eger (Hungary). E-mail: kiraly.sandor@uni-eszterhazy.hu

Tamás BALLA, Eszterházy Károly University, Faculty of Informatics, Eger (Hungary). E-mail: balla.tamas@uni-eszterhazy.hu