Journal of Positive School Psychology http://journalppw.com
2022, Vol. 6, No. 7, 4373-4401

Finding The Best Approach For Using Serious Games In Teaching Computer Programming

Abdelbaset Jamal Assaf[a], Barry McCollum[b] and Paul McMullan[b]

[a]*Luminus Technical University College, Jordan*
[b]*Queen's University Belfast, UK*

Abstract

Several studies reported the problem of high attrition and failure rates in computing schools. Many solutions were proposed and used to overcome these difficulties. The use of serious games for teaching computer programming was among the proposed solution. Several studies have reported the use of serious games and the benefits of using such educational method. However, different approaches can be applied when using serious games and no study compared the different approaches to find the best way for using serious games. This study compares two different approaches for using a serious game called Robocode for teaching computer programming. The study conducted an experiment with first year students taking introductory to programming course in three universities in Jordan. The results showed that using serious games for teaching computer programming through tutorials is the best approach. The results showed a significant improvement for using serious games through tutorials when compared to not using serious games at all.

1. Introduction

Technology has shaped and reformed almost every aspect of our life over the past decades. We are currently living in a digital world and Computer Science has a huge impact on this world. Thus, the demand on computer scientists is increasing to cope with the needs of this growing industry; however, computing faculties are facing major problems embodied by the high attrition rates and high failure rates Bennedsen and Caspersen (2019); Sithole, Chiyaka, McCarthy, Mupinga, Bucklein and Kibirige (2017); Dasuki and Quaye (2016); Swamidurai and Kannan (2016); Seyal, Mey, Matusin, Siau and Rahman (2015); Watson and Li (2014); Corney, Teague and Thomas (2010); Lu and Fletcher (2009); Gomes and Mendes (2007); Kinnunen and Malmi (2006); Beaubouef and Mason (2005). Simon, Luxton- Reilly, Ajanovski, Fouh, Gonsalvez, Leinonen, Parkinson, Poole and Thota (2019) "found that pass rates in introductory programming courses appear to average about 75%; that there is some evidence that they sit at the low end of the range of pass rates in introductory STEM courses.". Furthermore, Swamidurai and Kannan (2016) stated that the average failure rate at Alabama State University is about 30% to 35% and the failure rate of various computer programming courses are very high when compared to non-programming courses.

Computer Science majors have been facing a sharp decline in enrolment Ali (2009); Benokraitis, Bizot, Brown and Martens (2009). The Taulbee survey informed that about 50% fewer students entered Computer Science in 2007 compared to 2000 Zweben (2008). The Taulbee

survey is a survey conducted yearly by the Computing Research Association to document trends in student enrolment and degree production in the United States and Canada universities that offers PhD in Computer Science, Computer Engineering or Information. The first increase in six years in enrolment for computing majors in the USA happened in 2008 with an increase of 6.2% compared to 2007 Zweben (2009). An increase for enrolments in computing majors has been reported each year by the Taulbee survey with the last survey reporting an increase by 11.4% in 2017 comparing to the previous year Zweben and Bizot (2017). The survey reports a decrease of 12% in the awarding of BSc degrees in computing majors in 2009 compared to 2008 Zweben (2010). However, from 2010 to 2017 there was an increase in the awarding of the degrees. The increase of the awarding rate was expected because of the increase in the enrolments for the surveyed universities. But the survey does not report the attrition that occurs during the years. Moreover, the survey depends on the number of responses it receives. Each year, new universities are included in the survey and sometimes the previous universities can choose not to report. This makes the results inconsistent because some universities can simply not report to the survey when they have low enrolment or high failure rates. This can affect the results of the survey.

In 2010/2011, Woodfield (2014) found that Computer Science was the discipline with the lowest continuation rate
(91%), while students taking Computer Science formed 4.2% of the whole student body (67,847 people) included in the study. Similarly, University of West England reported an extremely low continuation rate in all computing programs. In the 2010/2011 academic year, the continuation rate was 78% and 82% in 2013/2014 Green, Plant and Chan (2016). Further, the study reported the retention or continuation rate for all the programs in the university between the years 2010 to 2015 and the continuation rate was more than 90% for all the years. Also, a study carried out by Talton, Peterson, Kamin, Israel and Al-Muhtadi (2006) stated that about 25% of the total number of entering freshmen have dropped out of the Computer Science program by the end of their first year from 1998 to 2003. The failure and dropout rates are high, mostly in the introductory computer programming courses Gomes and Mendes (2007). Beaubouef and Mason (2005) identified that there is about 30%-40% attrition rate for computing students and the vast majority of that occurs after the introductory to programming module. Also, Mcdowell, Werner, Bullock and Fernald (2006) stated that students change their majors after taking the first programming course in Computer Science. It has been shown that the problem occurs during the introductory to programming module, which leads to low continuation rate. Beaubouef and Mason (2005) connected the high attrition rate to poor advising, poorly planned labs, poor problem solving and math skills and the lack of practice and feedback. Moreover, some studies Manaris (2007); Beaubouef and Mason (2005) have stated that the complexity of the programming languages used in the introductory courses might be one of the factors affecting the attrition rate. Furthermore, Gomes and Mendes (2007) have highlighted the need for an increased amount of practice time and students' engagement.

1.1. Difficulties in teaching computer programming

While investigating the high attrition rate in computing it has been found that computer programming attracted more interest over other Computer Science topics. Bergin and Reilly (2005) have confirmed the difficulties in learning computerprogrammingandhaveconcludedthatthi scanresultinhighattritionandfailurerates.

According to Azadand Shubra (2010), the study estimated at least 25% of students drop the introductory to programming course because of the difficulty in learning computer programming. In the recent years, numerous studies have emerged to demonstrate that novice programmers have difficulties in learning Object-Oriented Programming (OOP) concepts Kunkle and Allen (2016); Biju (2012); Bennedsen and Caspersen (2007); Goosen and Pieterse (2005); Kelleher and Pausch (2005); Ragonis and Ben-Ari (2005); Hanks, Mcdowell, Draper and Krnjajic (2004); Robins, Rountree and Rountree (2003); Jenkins (2002). For example, students face several problems understanding classes, objects, recursion and inheritance Yan (2009). There are issues that emerge when teaching programming at an early stage, where students struggle with analyzing and designing of the code Papadopoulos and Tegos (2012); Dann, Cooper and Pausch (2000); Lopez, Whalley, Robbins and Lister (2008); Cooper and Pausch (2000). Further, students face difficulties because of the rigid programming syntax and the large amount of time required to assemble a simple output Olipas (2022); Sloan and Troy (2008); Wilson (2002). Moreover, Qian and Lehman (2017) expressed that students have difficulties in syntactic knowledge, conceptual knowledge, and strategic knowledge.

1.2. Innovative ways of teaching programming

Lately, there has been criticism of the traditional methods of teaching computer programming implying that they don't allow students to apply their knowledge, which leads to a decrease in student retention. Queirós, Pinto and Terroso (2020); Bosse and Gerosa (2017) stated that teaching computer programming is a complex activity that requires a lot of practice and traditional teaching approaches have not been able to respond effectively. Moreover, Zhang, Zhang, Stafford and Zhang (2013b) reported that traditional and conventional teaching methods using static material such as books and slides no longer relevant and effective in teaching and learning of computer programming. Thus, Cheah (2019) elucidated that teaching material should be able to support spatial visualization by incorporating multimedia elements and interaction to explain the dynamic concept of programming.

Several studies have claimed that academic advising, involvement and engagement, well-prepared teaching material, student support services and learning experiences are vital for student retention Roberts and Styron (2011); Barker, McDowell and Kalahar (2009); Michael and Chen (2005). Further, Cheah (2020) highlighted four factors which contribute to the difficulties in teaching and learning computer programming, which are phases of programming stage, problem solving skills, ineffective pedagogy and personal traits and attitude. Also, lack of understanding during the early stage of computer programming is one of the main factors contributing to the failure to understand computer programming Piwek and Savage (2020); Savage and Piwek (2019); Bosse and Gerosa (2017).

Thus, MacLean (2010) focused on the need for changing the introductory Computer Science courses teaching methods. Numerous proposed solutions were carried out. Some studies focused on students' attendance and marks, such as the one carried out by Green et al. (2016). They developed a system to identify and track students at risk of failing for the aim of preventing it. Some studies developed systems like the one in the study by Gálvez, Guzmán and Conejo (2009). They developed an Object-Oriented Programming System (OOPS), which is a problem-solving environment, where students

can solve OOP exercises and get instant on-demand feedback. Also, they used a web-based assessment system called SIETTE as a testing system to form their own blended e-learning approach. Other studies focused on improving the teachers' methods of delivering lectures\lessons such as the proposed teaching workshops by Porter, Lee, Simon and Guzdial (2017), which aimed to show how to be an effective teacher. Furthermore, the use of pair programming in introductory courses was implemented to contribute to greater perseverance in computing majors Porter, Guzdial, Mcdowell and Simon (2013); Mcdowell et al. (2006). Pair programming targets the problem of low retention rates that is caused by the low percentage of students' participation in activities. Sprint and Cook (2015), promoted group programming in a competition using decks of cards for questions.

2. Serious games in teaching computer programming

Serious games (SGs) have similar terms in the literature. Smith (2013) listed some of the popular terms, such as Educational Games, Edutainment, Simulation, Virtual Reality, Digital Game-Based Learning and Immersive Learning Simulations. The SGs studies are diverse in the literature and SGs have various definitions. Most of them describe and demonstrate SGs as interactive, entertaining, goal-focused and competitive Tobias and Fletcher (2007); Vogel and Wright (2006); E. and J. (1984). Abt (1970) provided one of the earliest definitions and descriptions of SGs, where SGs were described in these terms: "these games have an explicit and carefully thought-out educational purpose and are not intended to be played primarily for amusement. This doesn't mean that serious games are not, or should not be, entertaining". Alternative definitions for SGs are available in the literature such as Michael and Chen (2006) where they defined SGs as "a game in which education (in its various forms) is the primary goal, rather than entertainment". SGs offer educational content to users in an enjoyable way by simulating scenarios which promote learning. There are various definitions for SGs and its different terms, but they all agree on the importance of the delivered educational impact. Moreover, the definitions highlight the importance of different characteristics that must be present in the SGs to engage, motivate and immerse the users such as entertainment and enjoyment. If a SG doesn't engage or motivate the user in an interactive and entertaining way as a video game does, the user will not be immersed and focused while using the SG. Thus, the SG will fail to deliver its educational content to the user or the benefits of playing the game will be minimized.

As a result, the Computer Science teachers are using SGs to stimulate students Ramabu, Sanders and Schoeman (2021); ndrew Luxton-Reilly, Becker, Ott, Simon, Giannakos, Paterson, Albluwi, Kumar, Scott, Sheard and Szabo (2018); Corral, Balcells, Estévez, Moreno and Ramos (2014); Malliarakis and Xinogalos (2014); Eagle and Barnes (2009); Barnes, Richter, Powell, Chaffin and Godwin (2007).

3. Serious games usage and success factors

Several factors affect the success of using simulation software and SGs in teaching. First, learning environment, which should be interactive, flexible and personalised Malliarakis and Mozelius (2015); regardless delivered through lectures, lab sessions or assignments Sun, Tsai, Finger, Chen and Yeh (2008). Teo (2014) stated that learning environment is a key factor that influences the success of using e-

learning. Second, usage of space, the use of simulation software can overcome the problems associated with traditional learning related to space Navimipour and Zareie (2015); Tîrziu and Vrabie (2015);Wang (2014); Xu, Huang,Wang and Heales (2014). Third, students' access, whether limited or unlimited. Simulation software and SGs deliver 24/7 access to learning materials, which has a massive influence on the success of the learning process Zareie and Navimipour (2016); Omar, Hassan and Atan (2012); Gunasekaran, McNeil and Shaul (2002). Fourth, staff contacts' time, whether the use of simulation software and SGs requires extra staff time or not. Once the software is perceived to be easy to use, then no previous staff experience is required Capece and Campisi (2013); Rubin, Fernandes and Avgerinou (2013), and thus, no extra staff contact time. On the other hand, if the software is complicated then extra preparation is needed, which leads to extra staff time. However, the 21st-century students who are raised in a digital world are familiar with complicated technologies and computer games are part of their everyday life Malliarakis and Mozelius (2015). Fifth, cost, the use of SGs provides a cost-effective approach for reaching different learners and meeting continuous learning requirements Chen (2014); Lee, Hsieh and Ma (2011)). Learning environment is a crucial factor because it controls 4 other factors, which are staff contact time, students' access, usage of space and cost. Table 1 shows the controlled factors by learning environment. reaching different learners and meeting continuous learning requirements Chen (2014); Lee, Hsieh and Ma (2011)). Learning environment is a crucial factor because it controls 4 other factors, which are staff contact time, students' access, usage of space and cost. Table 1 shows the controlled factors by learning environment.

Table 1: The controlled factors by the learning environment factor

	Lectures	Assignments
Staff contacts' time	Time limited to usual lectures or extra lectures	Time to reply to students queries and questions
Students' access	Limited	Unlimited
Usage of space	Use of computer lab	No use of space
Cost	The cost of the serious game and running the lab	The cost of the serious game

Table 2 compares the several attempts of using SGs for teaching computer programming in terms of the country, learning environment, staff contact time, student access, usage of space and cost.

Based on the World Economic Situation and Prospects book United-Nations (2018), the countries were classified based on their economy. Figure 1 shows the percentage of the studies that were conducted in each country group. The figure shows that 75% of the studies were conducted in developed countries, in which there are no barriers in terms of the computing infrastructure or technology budget. Only 25% of the studies considered developing countries to test the use of SGs in teaching computer programming. While applying SGs in teaching computer programming in developing countries, several barriers appeared, such as the fragile computing infrastructure, lack of technology budget Talebian, Mohammadi and Rezvanfar (2014);

AlAmmary (2012) and cultural issues Farid, Ahmad, Niaz, Arif, Shamshirband and Khattak (2015). There are not enough studies and research to prove the possibility of the successful implementation and usage of SGs for teaching in developing countries. This formulates the need for more studies to investigate the feasibility and possibility of using SGs for teaching in developing countries. Furthermore, all the presented studies in table 2 concluded that SGs enhanced students' understanding of the covered topics and\or increased the motivation of the students. Using SGs can be done by assignments or lectures. Any of the two choices can affect and change several factors. Yet, no study compared the two approaches and investigated which approach achieves better results. This forms a gap that needs to be addressed, explored and analysed. Thus, there is a need for designing an experiment to investigate the effect of using SGs in teaching computer programming and to find the best approach to use such alternative. Further, there is a need to conduct the experiments in a developing country to investigate the feasibility and possibility of using SGs for teaching in developing countries. Designing and conducting the experiments will be answering the following research questions:

1. Does the use of serious games affect students' understanding of computer programming concepts?
2. What is the best approach for using serious games in teaching computer programming?
3. Is it possible to use serious games for teaching computer programming in developing countries?

Figure 1: Studies by country, See also Table 2.

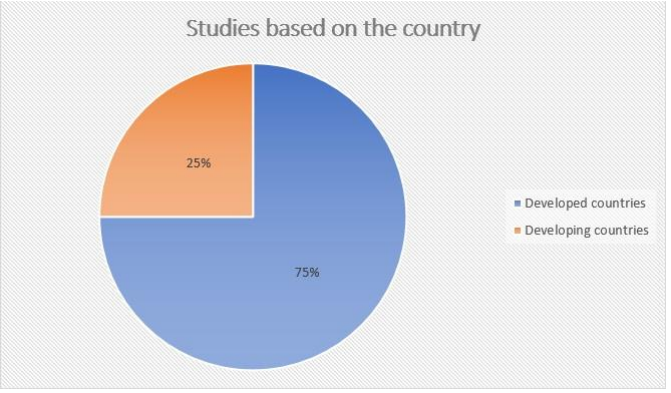


Table 2: Serious games studies summary

Study	Country	Learning environment	Staff contact time	Student access	Usage of space
Zhang,Caldwell and Smith (2013a)	USA	Tutorial alongside traditional lectures	Extra lectures	Limited access	Lab

Malliarakis and Xinogalos (2014)	Greece	Labs alongside traditional lectures	Extra lectures	Limited access	Lab
Corral et al. (2014)	Spain	Labs alongside traditional lectures	Extra lectures	Limited access	Lab
Eagle and Barnes (2008)	USA	Extra lectures after the semester	Extra lectures	Limited access	Lab
Ross (2002)	USA	Assignments	Regular lectures	Unlimited access	None
Baker, Zhang andCaldwell (2012)	USA	Labs alongside traditional lectures	Extra lectures	Limited access	Lab
Liu (2008)	Canada	Assignment	Regular lecture	Unlimited access	None
Al-Linjawi and Al-Nuaim (2010)	Saudi Arabia	Extra lectures	Extra lectures	Limited access	Lab
Hillyard, Angotti, Panitz, Sung, Nordlinger and Goldstein (2010)	USA	Assignment	Regular lectures	Unlimited access	None
Miljanovic and Bradbury (2017)	Canada	Assignment	Regular lectures	Unlimited access	None
Zhao and Muntean (2019)	Ireland	Labs alongside traditional lectures	Regular lectures	Limited access	Lab
Watson and Lipford (2019)	USA	Labs alongside traditional lectures	Regular lectures	Limited access	Lab
Agalbato and Loiacono (2018)	Italy	Assignment	Regular lectures	Unlimited access	None
Comber, Motschnig, Mayer and Haselberger (2019)	Austria	Labs alongside traditional lectures	Regular lectures	Limited access	Lab
Jordaan (2018)	South Africa	Labs alongside traditional lectures	Regular lectures	Limited access	Lab
Malliarakis and Xinogalos (2017)	Greece	Labs alongside traditional lectures	Extra lectures	Limited access	Lab

Erol and Kurt (2017)	Turkey	Labs alongside traditional lectures	Extra lectures	Limited access	Lab
Topalli and Cagiltay (2018)	Turkey	Labs alongside traditional lectures	Regular lectures	Limited access	Lab
Zhao, Chis, Muntean and Muntean (2018)	3 European countries	Labs alongside traditional lectures	Extra lectures	Limited access	Lab
Mathrani and Ponder-Sutton (2016)	New Zealand	Labs alongside traditional lectures	Regular lectures	Limited access	Lab
Ouahbi, Kaddari, Darhmaoui, Elachqar and Lahmine (2015)	Morocco	Labs alongside traditional lectures	Extra lectures	Limited access	Lab
Rozali and Zaid (2017)	Malaysia	Labs alongside traditional lectures	Extra lectures	Limited access	Lab
Rajeev and Sharma (2018)	USA	Labs alongside traditional lectures	Extra lectures	Limited access	Lab
Galgouranas and Xinogalos (2018)	Greece	Labs alongside traditional lectures	Extra lectures	Limited access	Lab

4. Methodology

This section and the following sub-sections will describe the chosen students sample, the experimental design and tasks of the experiment, the used SG and the data analysis technique that will be used.

4.1. Students sample

Recruiting participants is a challenge, but since our target is university students, announcements will be made in the lectures about the experiment. The announcement day will be after the teachers covered specific programming concepts and not at the beginning of the semester. Because students will get distracted from all the information they will get at the start of the semester and to ensure that all the students understand that this experiment and its tests are independent of the module itself. After obtaining ethical approval, the experiment will be described for all the students on the announcement day highlighting all the benefits students can gain from participating. Students who want to participate will read and sign a consent form that describes the experiment from the beginning to the end.

To decide who to recruit, the population must be analyzed. Since the study focuses on teaching programming concepts, we need to check the

population that want to participate in the experiment in terms of any previous programming experience and any experience in using the SG that we will use. Students with previous programming experience apart from high school programming materials and any previous knowledge of the chosen serious game will be excluded. The study Feigenspan, Kastner, Liebig, Apel and Hanenberg (2012) found that self-estimation of programming language experience for undergraduate students correlates with programming tasks. Thus, participants will be asked about their programming experience and about any familiarity with the chosen SG before including them in the experiment.

With regards to the student sample and as Singh (2006) stated "the size of the sample depends upon the precision the researcher desires in estimating the population parameter at a particular confidence level. There is no single rule that can be used to determine sample size. The best answer to the question of size is to use as large a sample as possible". Moreover, the study by All, Castellar and Looy (2016) conducted a semi-structured interview with experts who were defined as "staff members of an organization with a specific professional function and a specific experience and knowledge for this purpose". The experts were chosen to have at least a PhD degree or still conducting relevant research, which evaluates educational interventions. The aim of the interviews was to define the preferred methods for conducting digital game-based learning effectiveness studies. The results reported that an absolute minimum suggested by the experts is 20 participants per condition. Therefore, the study aims to recruit as many students as possible and ensure that a minimum of 20 participants is allocated to each group.

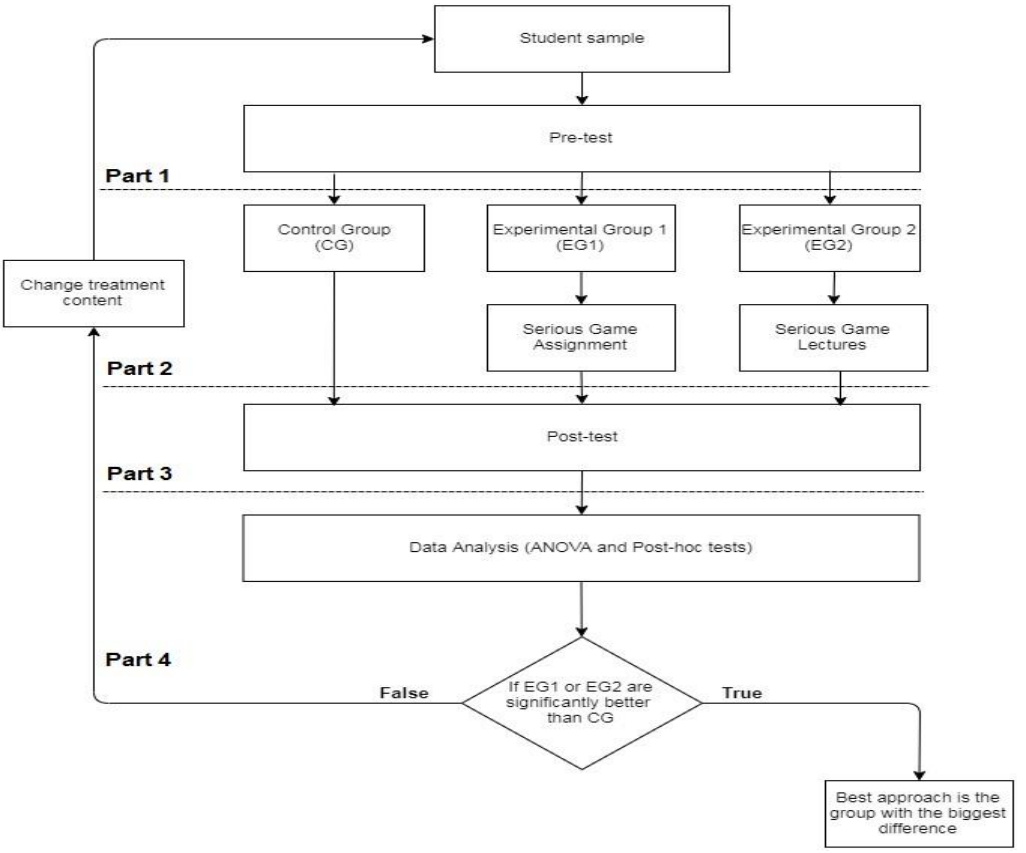
4.2. Experimental design

The chart in figure 2 presents the study experiment that follows a Comparative design, which is used to compare the effectiveness of different treatment modalities Kumar (1996). For example, to compare the effectiveness of three teaching models (A, B and C) on the level of comprehension of students in a class. The experiment design consists of three groups as follow:

1. The Control Group (CG): students will only take Pre- and Post-Tests.
2. The First Experiment Group (EG1): students will take the Pre- and Post-Tests. Students in this group will take an induction lecture followed by an assignment, where they will use the serious game to learn and apply their knowledge of programming. The group will take 2 weeks to complete the assignment.
3. The Second Experiment Group (EG2): students will take the Pre- and Post-Tests. Students in this group will take lectures, where they will be guided to use the serious game to finish the same assignment as students in EG1. The group will take 4 lectures, 2 hours each to complete the assignment.

All et al. (2016) conducted interviews with experts to define the preferred methods for conducting digital game based learning effectiveness studies. The results of the study showed that randomization has been accepted by all experts as the preferred method for assigning the participants to condition. However, Matching has been suggested by most of the experts (12 out of 13) as a method to guaranteeing similarity between conditions and controlling for certain variables. Table 3 provides an overview of variables to match participants in different conditions as suggested by the experts.

Figure 2: Research experiment



The students' distribution in the groups will be based on their previous knowledge represented by the pre-test scores. Students will be divided equally into the three groups according to their scores instead of randomly distributing them to avoid bias. Part 1 in figure 2 shows that the chosen student sample will take a pre-test and based on the results, the students will be divided into the three groups shown in part 2 of the figure.

Part 3 of figure 2 shows that the students will take a post-test. After completing the test and as shown in part 4 of figure 2, the results will be analyzed using ANOVA and post-hoc tests. Since we have two experimental groups and one controlled group, and the data collection will be based on a between-subject design, where different groups of people are assigned to each group. The tasks that will be assigned to the groups are different from one group to another. But the common task will be the pre-post-tests. Students in all three groups will take the same tests. The EG1 involves playing a serious game as an assignment, where students will be given a short induction lecture and they will be provided with the required tools to complete the task. They will be given 2 weeks to complete their task. The students in this group will have a 24/7 access to the serious game either from the university or from their personal computer devices. On the other hand, the EG2 is limited to 4 lectures, 2 hours each, where students will be guided on how to use the serious game to complete the required task. The students will have access to the serious game through the university computers only and have limited hours. Table 4 shows the location, covered topics and the duration for the experimental groups.

Table 3: Variables suggested to match on.

Variables	Description
Previous knowledge	Matching on prior academic achievement or pre-test scores
Ability	Matching on different ability levels (e.g. Low, medium and high achievers)
Motivation	Matching on motivation towards the learning content
Game experience	Matching on previous experience with games
Gender	Matching on gender (male\female)
Age	Matching on age\age categories
SES	Matching on socio-economic status

Table 4 Task design.

	Experiment group 1 (EG1)	Experiment group 2 (EG2)
Location	Any	University lab
Covered topics	Variables, methods, objects, strings, loops and arrays	Variables, methods, objects, strings, loops and arrays
Duration	2 weeks	4 lectures, 2 hours each

4.3. **Pre- and post- tests**

The pre- and post-tests will be used to measure students understanding of a specific computer programming concepts, such as variables declaration, methods calling, strings, loops and arrays. Both tests are formed of 20 multiple choice questions each and designed using g Alliger and Horowitz (1989) concept of multiple-choice tests to eliminate the guessing from changing the score of the test. Each question has a subsequent question asking if the student knows the answer or the student is just guessing. Figure 3 shows an example. In the pre- and post-tests, the questions are divided as follow:

1. The first 2 questions are about variables and methods declaration.
2. Questions (3, 4, and 6) are about calling methods.
3. Question 5 is about creating an object.
4. Questions (7, 8, 9, 10, and 11) are about Strings and its methods.
5. Questions (12, 13, 14, 15, and 16) are about Loops.
6. Questions (17, 18, 19, and 20) are about Arrays.

Using this concept in the tests allows to eliminate the guessing factor or knowing the correct answer by luck. Also, by using this concept we can get the answers in the traditional way by just marking the right answer or we can add the filter "Yes, I know the answer" so we can get the mark

based on the actual understanding of the topic that is being tested. Moreover, using this concept can reveal the misunderstanding of a certain topic when the answer is wrong but marked as "Yes, I know the answer". Another factor can be identified and analyzed, which is either the increase or the decrease of confidence in the knowledge when answering the questions between the pre-and the post-tests.

Figure 3: Question example

1. Which of the following is true about variables declared as final?

a. They cannot be initialised	b. They must be private
c. Their value cannot be changed	d. They cannot be used in a method

Yes, I know the Answer		No, I am guessing	

4.4. Variables

We have one Independent Variable (IV) with three levels and each level is assigned to a group. The CG has class lectures as the first level. The EG1 has the serious game assignment along with the class lectures as the second level. The EG2 has the extra serious game lectures along with the class lectures as the third level. Table 5 shows the three IV levels, their assigned group and their allocated task.

Table 5: Independent variable levels.

Independent variable levels	Group	Task
Level 1	Control group (CG)	Attend class lectures
Level 2	First experimental group (EG1)	Attend class lectures and take the serious game assignment
Level 3	Second experimental group (EG2)	Attend the class lectures and extra serious game lectures

The study has one dependent variable (DV), which is students' understanding that will be measured by the marks of the post-test. Students in all three groups will take the same test at the end of the experiment as shown in part 3 in figure 2.

4.5. The used serious game

The selected SG is Robocode which is short for "Robot Code". It is an open-source Java-based virtual robot game that is intended to teach object-oriented programming concepts. The Robocode game consists of a robot-development tool and it simulates a virtual battlefield where robots can battle against each other. The player programs the robot commanding it how to perform and respond to events arising in the battlefield. Thus, Robocode forms a space for students and learners to learn and apply their knowledge in OOP. It covers writing classes, reading, analyzing and using existed code, event handling and message passing Bonakdarian and L. (2004). Robocode battles are running in real-time and on-screen. The game starts at least with

two robots and each one of them starts with the energy of 100 and dies when it drops to zero. The game ends when there is only one robot left on the battlefield or when the time runs out. Robot class is automatically generated when creating a new robot and the class contains a run method with an infinite loop that defines the default behavior for the robot. Additionally, the object has methods such as onScannedRobot, onHitWall, and onHitByBullet which are responsible for handling a response to a particular event by calling other methods which will perform some actions either by moving the robot or investigating about the opponent robot.

What makes Robocode a good game for learning is that it represents the object with a visual activity. The students or learners can see the results and consequences of their implementation and calculations live on the battlefield. Furthermore, the rules of the game in terms of losing and gaining energy requires deep thinking and push students or learners to use different and analyzed strategies. Because when a robot fires a bullet, the same robot loses energy with the same amount of the firepower, which makes the game not only about firing bullets. However, if the bullet hits another robot, the robot will gain back some energy and the opponent tank will lose four times the firepower. Hence, making a decent robot requires such an implementation that effectively fires and dodges bullets. A notable feature of Robocode is that it allows the users to instantly view and test their robots' behavior against different provided sample robots. This makes the testing and debugging easy and interactive. Thus, making the game covers several stages of the software development process. Moreover, the battling and challenging feature of the game provides a competitive and fun factor that acts as an attraction feature for the students. The game acts as a motivational element to impulse the students to study and understand different programming concepts to be able to create a robust robot, which will lead into adding to the overall students' experience. Robocode creator Mat Nelson stated that Robocode is "like chess, simple to learn, difficult to master" Triplett (2002), since a simple robot can be developed in minutes, where a sophisticated robot can take months of development. This means that Robocode can be used on various levels of students and even on experienced Computer Science graduates. Further, Hartness (2004) stated that Robocode can be used to encourage students to master difficult concepts and apply their knowledge in an interesting situation. Also, they can use Robocode to apply their knowledge without the need for mastering all the details of the game.

4.6. Data analysis

Since we want to compare the mean of different groups. T-test can be used to compare the different groups in pairs and repeat this process to test the multiple groups. However, using multiple t-test comparison is not appropriate statistical practice. Because the chance of committing a type I error (false positive, rejecting the null hypothesis when it is true) is high Weaver, Morales, Dunn, Godde and Weaver (2017). Instead of using multiple t-test comparison, Analysis of Variance (ANOVA) can be used to compare the mean of multiple groups. For example, the study Ashby, Sadera and McNary (2011) used ANOVA to compare the student success between developmental math courses offered online, blended and face-to-face. Since we have three different groups, ANOVA will be used to compare the effect of the method of learning programming on the students understanding represented by the multiple-choice post-test they will take. Weaver et al. (2017) listed five assumptions that must be satisfied before using ANOVA, which are:

1. Data type: “The dependent variable must be interval or ratio. Additionally, the independent variable should have two or more categorical groups”. This means that the dependent variable must be continuous, such as height measured by cm or exam scores measured by numbers. The independent variable represents the independent groups and normally ANOVA is used when there are three or more categories or groups, such as ethnicity or treatment type. ANOVA can be used for two groups but using t-test is more common in this situation.
2. Distribution of data: “The data follow a normal distribution. This includes the dependent variable’s distribution within each category (group) of the independent variable”. This means that a test for data normality must be applied on the dependent variable for each independent variable level separately. The one-way-ANOVA is robust to violations of normality, which means that this assumption can be a little violated and still provide valid results.
3. Independent samples: “The samples are independent (with the exception of rANOVA); independent samples have no effect on one another”. It can be also referred as independence of observations, which means that there must be different participants in each group with no participant being in more than one group. This assumption is a matter of design, in which using a between-subject design will satisfy this assumption. The between-subject design means that different groups of people are assigned to each group unlike the with-in subject design were the same group will do all the tasks.
4. Homogeneity: “The variance of the populations must be equal, meaning the populations must have an equal spread around the mean”. This means that the groups’ variances must be homogeneous. This assumption can be checked using Levene’s test for homogeneity of variances.
5. Random sampling: “Theobservationsarerandomlysampledandareindependentfromoneanother”. This means that each data point in the population has an equal chance of being included in the sample. The independent samples part has been described in point 3.

If all the previous assumption were met and ANOVA was used. The result will only show if there is a significant difference in the mean between the tested groups or not. If the result showed that there is a significant difference in the mean. It will not determine which group is statistically different from the other. In order to specify where the differences lie, there is a need to use post-hoc tests, such as Tukey honestly significant test (HSD) Weaver et al. (2017).

5. Results and discussion

The experiment was conducted in universities in Jordan as a case study of a developing country. The experiment was conducted in Applied Science University, Petra University and the University of Jordan. The experiment aimed to investigate the positive impact of using simulation software and SGs for teaching computer programming and to identify the best approach of using this method in teaching to increase the potential outcomes. Also, to investigate the feasibility and possibility of using SGs for teaching in developing countries.

Meetings with the coordinators of the programming courses for Year 1 students took a place in three universities. Topics, such as how to interact with students and how to announce the competition were discussed. Permissions were granted from the teachers of the programming modules for year 1 students in all majors to make a presentation in the lectures to introduce the students to the competition. Interested students who want to participate were asked to sign in a consent form. 123 out of roughly 400 students showed interest and signed in for the experiment, in which 81 were males and 42 were females. Following the framework, a pre-test has been conducted to evaluate the students' programming skills and knowledge and it was given to all available year 1 students. 174 students completed the pre-test to ensure we have a control group that matches the two experimental groups.

Following the framework and as shown in Part 1 in figure 2. Students who signed for the experiment were divided into two groups equally based on their marks in the test to ensure that each group has students on the same level. Then students were informed, in which group they are and the students in the first experimental group who meant to work on their own were given access to a website, where they can download the software development kit they need to build their own robots and to use it for submitting the final work. Also, the students were given a document, which acts as a short tutorial to help them start with the game. The students in the second experimental group who meant to take tutorials to build their robots were informed of the times that a computer laboratory was booked to start developing the robots. The structure of the tutorials is as follow:

1. Tutorial 1: The students were welcomed and guided through the steps to download the Robocode software development kit and install in on the machines. Then, the students were shown how to start the game and how to do simple and basic stuff, such as starting a battle and creating new robots. The students were asked to change the initial code, which is automatically generated when a new robot is created and see the results of the changes by playing the game. Some rules were explained to the students, such as the rules of firing bullets and the energy calculation, information about the battlefield and its coordination, the movement methods and its parameters and the anatomy of the robots, which consists of three main parts, which are body, gun and radar.
2. Tutorial 2: Reaching the second tutorial, the students were already familiar with the basics of Robocode so they were introduced to the events in the game and how can the events be used and what data can be retrieved from the occurrence of a certain event. The students were introduced to the getters, in which they can retrieve information about other robots in the battlefield, such as the robot’s energy, distance and X and Y-axis. The students were then asked to start targeting the other robots by calculating the distance between the two robots and accordingly assign a suitable firepower to the fired bullets.
3. Tutorial 3: In the third tutorial, the students were asked to create their own robot behavior and to draw a design. After that, the students started developing their robots based on the created design. During the development, the students were given assistance and guidance to help them in completing the objectives.
4. Tutorial 4: During the final tutorial, the students continued the work on their

robot's design and at the end of the tutorial the students tried their robots against each other to help them tune their robots. Finally, the students submitted their final work using the provided website.

43 students (29 male and 14 female) from the two experimental groups completed the tasks, submitted their final work and completed the post-test. Another 20 students (14 male and 7 female) who represent the control group completed the post-test. The small number of students who completed the experiment can be justified by the timing of the experiment, in which most of the students who drooped from the experiment referred that to the timing of experiment being at the end of the semester, in which the students had many assignments and assessments. However, the timing of the experiment can't be changed, because students must cover different programming concepts before getting involved in the experiment.

5.1. Results and discussion

ANOVA statistical test was chosen, and ANOVA five assumptions must me be satisfied before using ANOVA. The assumptions will be checked and make sure they are all met as shown below:

1. Data type: The dependent variable data must be interval or ratio. The dependent variable data we have is interval, which is represented by the exam scores.
2. Distribution of data: The data must be tested to check if it is normally distributed. Shapiro-Wilk test was chosen as a test for data normality because several studies stated that Shapiro-Wilk test is the most powerful normality test, such as Farrell and Rogers-Stewart (2006); Keskin (2005). In Shapiro-Wilk test, a null hypothesis H_0 means the population is normally distributed and the alternative hypothesis H_1 means the population is not normally distributed. If the significant value obtained from the results is greater than 0.05 that means that the data came from a normally distributed population and if the significant value was less than 0.05 that means the null hypothesis is rejected and the sample population is not normally distributed. Table 6 shows the results of applying Shapiro-Wilk test on all three groups.

As shown in table 6, the significance value for all three groups is greater than 0.05, which means we accept the H_0 that the data came from a normally distributed population.

Table 6 Shapiro-Wilk test results.

Normality test	Shapiro-Wilk test	
	Statistic value	Significant (P-value)
Control group	0. 969	0. 725
First experimental group	0. 979	0. 904
Second experimental group	0. 960	0. 552

3. Independent samples: The research design followed the between-subject design, in which different groups of people are assigned to each group. Thus, there are different participants in each group, which means the samples are independent.
4. Homogeneity: To check this assumption, Levene's test of homogeneity was used. The original Levene's test used only the mean. Brown and Forsythe (1974) extended the test to use the median or 10% trimmed mean. The 10% trimmed mean is the mean of the observations after removing the largest and smallest 10% values in that group. Levene's test null and alternative hypotheses are defined as:
    - H_0: The data samples have equal variances.
    - H_1: The data samples don't have equal variances.

    Levene's test was applied on the data using the mean, median and 10% trimmed mean to test the hypothesis that the groups' variances are equal. Below are the results:
        a) Use of mean: The p-value was 0.6686.
        b) Use of median: The p-value was 0.7258.
        c) Use of 10% trimmed mean: The p-value was 0.6725.

    All three tests failed to reject the null hypothesis at the 0.05 significance level. There is insufficient evidence to claim that the variances are not equal and, thus, this assumption for using ANOVA is satisfied.
5. Random sampling: The samples were randomly selected and were randomly assigned to the groups. The matching in the groups was based on the marks but the process of assigning the participants to the groups was done randomly.

After all five assumptions were satisfied, ANOVA will be applied on the results of the post-test to test if there are any differences between the three groups. Sullivan and Feinn (2012) defined effect size as "the magnitude of the difference between groups". It helps readers understand the magnitude of differences found. The p-value can inform the reader whether an effect exists while the effect size will reveal the size of the intervention effect. For illustrative purposes, effect sizes were calculated using the sample size, mean and variance of the three groups. The results showed the following:

1. The effect size for EG1 vs EG2 is f=0.2288
2. The effect size for EG1 vs CG is f=0.4343
3. The effect size for EG2 vs CG is f=0.2178
4. The overall effect size is f=0.4343

Sullivan and Feinn (2012 defined statistical power as "the probability that your study will find a statistically significant difference between interventions when an actual difference does exist". If statistical power is high, the likelihood of deciding there is an effect, when one does exist, is high. The power was calculated using the achieved overall effect size, which is 0.4343 and with the significance level set to 0.05. The result showed that the power is 0.8683 which means that the significant results are reliable due to the reduction in probability of type II error.

ANOVA is used to determine whether there are any statistically significant differences between the means of groups. ANOVA generates two hypotheses, which are:

- H_0: There are no statistically significant differences between the means of the groups.
- H_1: There are statistically significant differences between the means of the groups.

Running ANOVA showed that the effect of teaching approach on students results in the post-test was significant, F (2, 61) = 6.01, p = 0.004. This means we reject the null hypothesis H_0 and accept the alternative hypothesis H_1. Table 7 shows the mean and standard deviation for all three groups. Table 8 shows the generated table from running ANOVA and figure 4 shows the ANOVA boxplot.

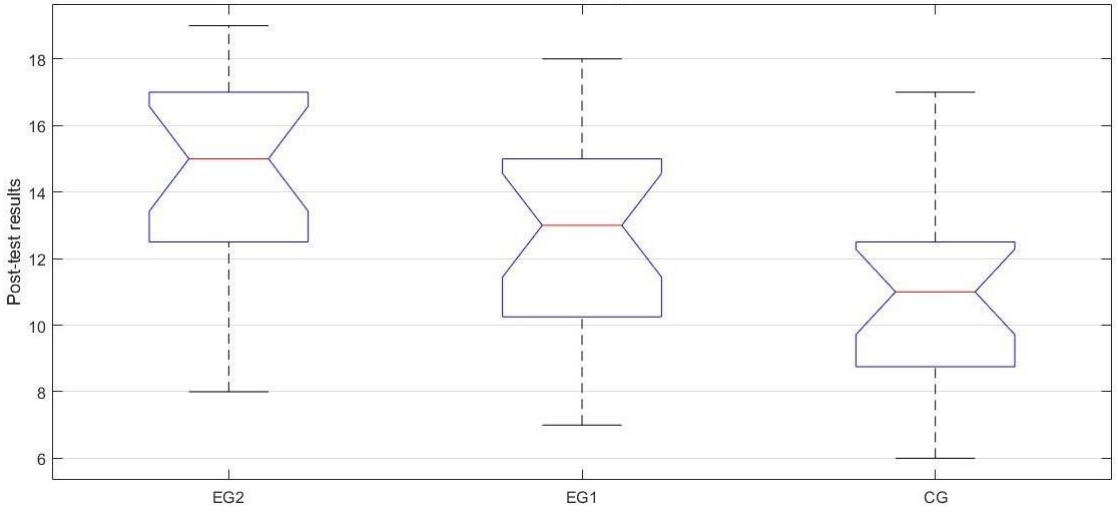
Table 7: Groups mean and standard deviation.

	Experimental group 1	Experimental group 2	Control group
Mean	12.695	14.5	11
SD	3.308	3.12	3.245

Table 8: Result of running ANOVA

	Sum of squares	Df	Mean square	F	Sig.
Between groups	125.4898	2	62.7449	6.019	0.004
Within groups	635.8696	61	10.42409		
Total	761.3594	63			

Figure 4: ANOVA boxplot.



ANOVA test will only show that there is a significant difference in the mean. It will not determine which group is statistically different from the other. In order to specify where the

differences lie, there is a need to use post-hoc tests, such as Tukey honestly significant test (HSD) Weaver et al. (2017). HSD was used as a post-hoc test to investigate where the differences occurred between the three groups. HSD is designed to compare each of the conditions (groups) to every other condition (group). Thus, it will compare EG1 with EG2, EG1 with CG and EG2 with CG. HSD will run three times and each time it generates two hypotheses, which are:

- H_0: There are no statistically significant differences between group 1 and group 2.
- H_1: There are statistically significant difference between group 1 and group 2.

Running HSD showed the following:

- EG2 and CG differed significantly at $p<.05$, in which the p-value was 0.002. Thus, H_0 is rejected and the alternative hypothesis H_1 is accepted.
- EG2 and EG1 were not significantly different, in which the p-value was 0.16 and, thus, the null hypothesis H_0 is accepted.
- EG1 and CG were not significantly different, in which the p-value was 0.19 and, thus, the null hypothesis H_0 is accepted.

The statistical test ANOVA showed that there is a statistically significant difference between the three groups. When HSD was applied to investigate further. The results showed that students in the second experimental group, where students used the SG and took the tutorials (M=14.5, SD=3.12) achieved significantly higher marks in the post-test compared to the control group, where students didn't use the serious game (M=11, SD=3.24). HSD didn't identify any significant difference between the students in the second experimental group and students in the first experimental group (M=12.695, SD=3.308). Also, the results showed that there is no significant difference between the students in the first experimental group and students in the control group.

The statistical analysis shows that the test results are significantly different for the three groups. The significant difference occurred between the control group and the group that used the serious game through tutorials. The difference between the other groups is big but not significant. Considering the test results, which represents students' understanding of the covered concepts. It can be concluded that using SGs through tutorials is the best approach for using SGs in teaching computer programming. Furthermore, conducting the experiment in Jordan as a case study of a developing country showed that using serious games for teaching in developing countries is possible. The barriers represented by the fragile computing infrastructure and the lack of technology budget didn't affect the possibility and feasibility of using this approach in teaching. Conducting the experiment in Jordan as a case study of a developing country answered all the research questions as follow:

1. Does the use of serious games affect students' understanding of computer programming concepts?

The data analyses showed that using SGs enhanced students' understanding in computer programming concepts.

2. What is the best approach for using serious games in teaching computer programming?

The data analyses showed that students' marks after using the SG through tutorials are significantly better than students' marks who didn't use the SG. Thus, this study concludes that using SGs through tutorials is the best approach for using this method in teaching.

3. Is it possible to use serious games for teaching computer programming in developing countries?

Conducting the experiment in Jordan as a case study of a developing country showed it is possible to use SGs for teaching in developed countries as the requirements for using this method of teaching are simple and are easily accessed.

6. Conclusion

The use of SGs was suggested by many studies as a solution for the difficulties in teaching computer programming, which could help in decreasing the attrition and failure rates. Numerous studies have used SGs for teaching computer programming. The results showed that SGs enhanced students' understanding of the covered concepts and\or increased the students' motivation. However, a gap was found since no study compared the different approaches of using SGs to increase the effectiveness of the used SG. This study compared two different approaches for using SGs in teaching, which are assignments and tutorials. Also, the study conducted the experiment in Jordan as a case study of a developing country to investigate the possibility and feasibility of using SGs in developing countries.

A framework was designed by combining five key factors for the success of using simulation software and SGs. The factors are learning environment, usage of space, students' access, staff contacts' time and cost. The framework consists of 2 experimental groups and 1 control group. The first experimental group used a serious game called Robocode to complete an assignment and they had unlimited access to the game. The second experimental group used the same serious game through lectures to complete the same assignment, but they had limited access to the game. The control group didn't use the serious game and they only attended the module lectures and labs. After completing the assignment, students in all groups completed a test to measure their understanding of the covered concepts. The results showed that using SGs through lectures is significantly better than not using SGs at all and it achieves better results in terms of students' understanding. Also, the study found it is possible to use SGs for teaching in developed countries.

Part of the future work is to conduct the same experiment but using different SGs to check if the same results will be achieved. Moreover, the study recommends using the framework and conducting the same experiment in other countries, both developing and developed to see if the same results can be obtained.

References

1. Abt, C., 1970. Serious Games. Lanham, MD: University Press of America 1987. doi:10.1177/000276427001400113.
2. Agalbato, F., Loiacono, D., 2018. Robo3: a puzzle game to learn coding, in: Proc. IEEE Games, Entertainment, Media Conference (GEM)., pp. 359–366. doi:10.1109/GEM.2018.8516515.
3. Al-Linjawi, A.A., Al-Nuaim, A.H., 2010. Using alice to teach novice programmers oop concepts. Journal of King Abdulaziz University-Science 22, 59–68. doi:10.4197/sci.22-1.4.
4. AlAmmary, J., 2012. Educational technology: A way to enhance students achievement at the university of bahrain, in: Procedia – Social and Behavioral Sciences., pp. 48–257. doi:10.1016/j.sbspro.2012.09.501.
5. Ali, A., 2009. Successful efforts in recruiting women into technology courses – a case study. Issues in Information System , 225–231.

6. All, A., Castellar, E.P., Looy, J., 2016. Assessing the effectiveness of digital game-based learning: Best practices. Computers Education 92-93,
7. 90–103. doi:10.1016/j.compedu.2015.10.007.
8. Alliger, G.M., Horowitz, H.M., 1989. Ibm takes the guessing out of testing. Training and Development Journal 43, 69–73.
9. Ashby, J., Sadera, W.A., McNary, S.W., 2011. Comparing student success between developmental math courses offered online, blended, and face-to-face. Journal of Interactive online Learning 10, 128–140.
10. Azad, A., Shubra, C., 2010. Efforts to reverse the trend of enrollment decline in computer science programs. Issues in Informing Science and
11. Information Technology 7, 209–224. doi:10.28945/1201.
12. Baker, A., Zhang, J., Caldwell, R.E., 2012. Reinforcing array and loop concepts through a game-like module, in: The 17th International Conference on Computer Games (CGAMES)., pp. 175–179. doi:10.1109/CGames.2012.6314572.
13. Barker, J.L., McDowell, C., Kalahar, K., 2009. Exploring factors that influence computer science introductory course students to persist in the major. ACM SIGCSE Bulletin 41, 153–157. doi:10.1145/1539024.1508923.
14. Barnes, T., Richter, H., Powell, E., Chaffin, A., Godwin, A., 2007. Game2learn: Building cs1 learning games for retention, in: The 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education., pp. 121–125. doi:10.1145/1268784.1268821.
15. Bayliss, J.D., 2007. The effects of games in cs1-3. Journal of Game Development 3, 7–17.
16. Beaubouef, T., Mason, J., 2005. Why the high attrition rate for computer science students. ACM SIGCSE Bulletin 37, 103–106. doi:10.1145/1083431.1083474.
17. Bennedsen, J., Caspersen, M.E., 2007. Failure rates in introductory programming. ACM SIGCSE Bulletin 39, 32–36. doi:10.1145/1272848.1272879.
18. Bennedsen, J., Caspersen, M.E., 2019. Failure rates in introductory programming: 12 years later. ACM Inroads 10, 30–36. URL: https://doi.org/10.1145/3324888, doi:10.1145/3324888.
19. Benokraitis, V., Bizot, B., Brown, R., Martens, J., 2009. Reasons for cs decline: Preliminary evidence. Journal of Computing Sciences in Colleges 24, 161–162.
20. Bergin, S., Reilly, R., 2005. The influence of motivation and comfort-level on learning to program, in: The 17th Annual Workshop of the Psychology of Programming Interest Group., pp. 293–304.
21. Betancur, A.J., Rodríguez, C., Ezparragoza, I., 2011. An undergraduate collaborative design experience among institutions in the americas, in: The 8th WSEAS International Conference on Engineering Education., pp. 263–267.
22. Biju, S.M., 2012. Difficulties in understanding object oriented programming concepts. Lecture Notes in Electrical Engineering Innovations and Advances in Computer, Information, Systems Sciences, and Engineering , 319–326doi:10.1007/978-1-4614-3535-

8_27. Bonakdarian, E., L., W., 2004. Robocode throughout the curriculum. Journal of Computing Sciences in Colleges 19, 311–313.
23. Bosse, Y., Gerosa, M.A., 2017. Why is programming so difficult to learn? patterns of difficulties related to programming learning mid-stage.
24. SIGSOFT Softw. Eng. Notes 41, 1–6. doi:10.1145/3011286.3011301.
25. Brown, M.B., Forsythe, A.B., 1974. Robust tests for the equality of variances. Journal of the American Statistical Association 69, 364–367. doi:10.2307/2285659.
26. Capece, G., Campisi, D., 2013. User satisfaction affecting the acceptance of an e-learning platform as a mean for the development of the human capital. Behaviour Information Technology 32, 335–343. doi:10.1080/0144929x.2011.630417.
27. Cheah, C.S., 2019. Screencasting: how effective is it in developing positive attitude towards the learning of c++ computer programming. Journal of Educational Sciences & Psychology 9.
28. Cheah, C.S., 2020. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. Contemporary Educational Technology 12. doi:https://doi.org/10.30935/cedtech/8247.
29. Chen, T.L., 2014. Exploring e-learning effectiveness perceptions of local government staff based on the diffusion of innovations model. Administration Society 46, 450–466. doi:10.1177/0095399713482313.
30. Comber, O., Motschnig, R., Mayer, H., Haselberger, D., 2019. Engaging students in computer science education through game development with unity, in: 2019 IEEE Global Engineering Education Conference (EDUCON), pp. 199–205. doi:10.1109/educon.2019.8725135.
31. Cooper, S., D.W., Pausch, R., 2000. Alice: A 3-d tool for introductory programming. Journal of Computing Sciences in Colleges 15, 199–205.
32. Corney, M., Teague, D., Thomas, R., 2010. Engaging students in programming, in: The Twelfth Australasian Conference on Computing Education., pp. 63–72.
33. Corral, R.M.J., Balcells, C.A., Estévez, M.A., Moreno, J.G., Ramos, F.J.M., 2014. A game-based approach to the teaching of object-oriented programming languages. Computers Education, 83–92. doi:10.1016/j.compedu.2013.12.013.
34. Dann, W., Cooper, S., Pausch, R., 2000. Making the connection: Programming with animated small world. ACM SIGCSE Bulletin 32, 41–44. doi:10.1145/353519.343070.
35. Daoudi, I., 2022. Learning analytics for enhancing the usability of serious games in formal education: A systematic literature review and research agenda. Education and Information Technologies. doi:10.1007/s10639-022-11087-4.
36. Dasuki, S., Quaye, A., 2016. Undergraduate students' failure in programming courses in institutions of higher education in developing countries: A nigerian perspective. The Electronic Journal of Information Systems in Developing Countries 76, 1–18. doi:10.1002/j.1681-4835.2016.tb00559.x. E., D.J., J., D.D., 1984. Microcomputer videogame based training. Educational Technology 24, 11–17.

37. Eagle, M., Barnes, T., 2008. Wu's castle: Teaching arrays and loops in a game. ACM SIGCSE Bulletin 40, 245–249. doi:10.1145/1597849. 1384337. Eagle, M., Barnes, T., 2009. Experimental evaluation of an educational game for improved learning in introductory computing. ACM SIGCSE Bulletin 41, 321–325. doi:10.1145/1539024.1508980.
38. Erol, O., Kurt, A.A., 2017. The effects of teaching programming with scratch on pre-service information technology teachers motivation and achievement. Computers in Human Behavior 77, 11–18. doi:10.1016/j.chb.2017.08.017.
39. Farid, S., Ahmad, R., Niaz, I.A., Arif, M., Shamshirband, S., Khattak, M.D., 2015. Identification and prioritization of critical issues for the promotion of e-learning in pakistan. Computers in Human Behavior 51, 161–171. doi:10.1016/j.chb.2015.04.037.
40. Farrell, P.J., Rogers-Stewart, K., 2006. Comprehensive study of tests for normality and symmetry: extending the spiegelhalter test. Journal of Statistical Computation and Simulation 76, 803–816. doi:10.1080/10629360500109023.
41. Feigenspan, J., Kastner, C., Liebig, J., Apel, S., Hanenberg, S., 2012. Measuring programming experience, in: 2012 20th IEEE International Conference on Program Comprehension (ICPC), p. 73–82. doi:10.1109/icpc.2012.6240511.
42. Galgouranas, S., Xinogalos, S., 2018. javant-garde: A cross-platform serious game for an introduction to programming with java. Simulation Gaming 49, 751–767. doi:10.1177/1046878118789976.
43. Gomes, A., Mendes, A.J.N., 2007. Learning to program-difficulties and solutions, in: The International Conference on Engineering Education, pp. 283–287.
44. Goosen, L., Pieterse, V., 2005. Goosen, l. and pieterse, v., (2005, in: The 35th conference of SACLA, pp. 283–287.
45. Green, S., Plant, N., Chan, C., 2016. Student at risk identification and remedial action system for improving retention on computer science programmes. New Directions in the Teaching of Physical Sciences doi:10.29311/ndtps.v0i11.572.
46. Gunasekaran, A., McNeil, R.D., Shaul, D., 2002. E-learning: research and applications. Industrial and Commercial Training 34, 44–53. doi:10.1108/00197850210417528.
47. Gálvez, J., Guzmán, E., Conejo, R., 2009. A blended e-learning experience in a course of object oriented programming fundamentals. Knowledge-Based Systems 22, 279–286. doi:10.1016/j.knosys.2009.01.004.
48. Hainey, T., Connolly, T.M., Boyle, E.A., Wilson, A., Razak, A., 2016. A systematic literature review of games-based learning empirical evidence in primary education. Computers & Education 102, 202–223. URL: https://www.sciencedirect.com/science/article/pii/S0360131516301567,doi:https://doi.org/10.1016/j.compedu.2016.09.001.
49. Hanks, B., Mcdowell, C., Draper, D., Krnjajic, M., 2004. Program quality with pair programming in cs1. ACM SIGCSE Bulletin 36, 176–180. doi:10.1145/1026487.1008043.
50. Hanzu-Paraza, R., Barsan, E., 2010. Teaching techniques –modern bridges

between lecturers and students, in: The 7th WSEAS International Conference on Engineering Education, pp. 176–181.

51. Hartness, K., 2004. Robocode: Using games to teach artificial intelligence. Journal of Computing Sciences in Colleges 36, 287–291.
52. Hillyard, C., Angotti, R., Panitz, M., Sung, K., Nordlinger, J., Goldstein, D., 2010. Game-themed programming assignments for faculty, in: The 41st ACM technical symposium on Computer science education - SIGCSE 10, pp. 270–274. doi:10.1145/1734263.1734358.
53. Hwang, G.J., Wu, P.H., 2011. Advancements and trends in digital game-based learning research: a review of publications in selected journals from 2001 to 2010. British Journal of Educational Technology 43, 6–10. doi:10.1111/j.1467-8535.2011.01242.x.
54. Jenkins, T., 2002. On the difficulty of learning to program, in: The 3rd Annual conference of the LTSN Centre for Information and Computer Sciences, pp. 53–58.
55. Jordaan, D.B., 2018. Board games in the computer science class to improve students' knowledge of the python programming language, in: 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC). doi:10.1109/iconic.2018.8601207.
56. Kasenides, N..P.N., 2021. amazechallenge: An interactive multiplayer game for learning to code, in: 29TH INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS DEVELOPMENT.
57. Kelleher, C., Pausch, R., 2005. Lowering the barriers to programming: A survey of programming environments and languages for novice programmers. ACM Computing Surveys (CSUR) , 83–137doi:10.1145/1089733.1089734.
58. Keskin, S., 2005. Comparison of several univariate normality tests regarding type i error rate and power of the test in simulation based small samples. Journal of Applied Science Research 2, 296–300.
59. Kinnunen, P., Malmi, L., 2006. Why students drop out cs1 course?, in: The 2006 international workshop on Computing education research – ICER 06, pp. 97–108. doi:10.1145/1151588.1151604.
60. Kumar, R., 1996. Research methodology: a step-by-step guide for beginners. Sage.
61. Kunkle, W.M., Allen, R.B., 2016. The impact of different teaching approaches and languages on student learning of introductory programming concepts. ACM Transactions on Computing Education 16, 1–26. doi:10.1145/2785807.
62. Lamb, R.L., Annetta, L., Firestone, J., Etopio, E., 2018. A meta-analysis with examination of moderators of student cognition, affect, and learning outcomes while using serious educational games, serious games, and simulations. Computers in Human Behavior 80, 158–167. doi:https://doi.org/10.1016/j.chb.2017.10.040.
63. Lee, Y.H., Hsieh, Y.C., Ma, C.Y., 2011. A model of organizational employees' e-learning systems acceptance. Knowledge-Based Systems 24, 355–366. doi:10.1016/j.knosys.2010.09.005.
64. Liu, P.L., 2008. Using open-source robocode as a java programming assignment. ACM SIGCSE Bulletin 40, 63–67. doi:10.1145/1473195. 1473222.
65. Lopez, M., Whalley, J., Robbins, P., Lister, R., 2008. Relationships between

reading, tracing and writing skills in introductory programming, in: The fourth international workshop on Computing education research - ICER 08, pp. 101–112. doi:10.1145/1404520.1404531.

66. Lu, J., Fletcher, G., 2009. Thinking about computational thinking. ACM SIGCSE Bulletin 41, 260–264. doi:10.1145/1539024.1508959.
67. Andrew Luxton-Reilly, Becker, B., Ott, L., Simon, Giannakos, M., Paterson, J., Albluwi, I., Kumar, A., Scott, M., Sheard, J., Szabo, C., 2018. Introductory programming: a systematic literature review, in: ITiCSE 2018 Companion - Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, Association for Computing Machinery (ACM). pp. 55–106. doi:10.1145/3293881.3295779.
68. MacLean, L., 2010. Recruitment and retention of women in computer science and information systems: How and why, in: The 2nd International Conference on Education and New Learning Technologies, pp. 1585–1591.
69. Malliarakis, C., S.M., Xinogalos, S., 2014. Cmx: Implementing an mmorpg for learning programming, in: The European Conference on Gamesbased Learning, pp. 346–355.
70. Malliarakis, C., S.M., Xinogalos, S., 2017. Cmx: The effects of an educational mmorpg on learning and teaching computer programming, in: IEEE Transactions on Learning Technologies, pp. 219–235.
71. Malliarakis, C., S.O., Mozelius, P., 2015. How to build an ineffective serious game: Worst practices in serious game design, in: The 9th European Conference on Games Based Learning, ECGBL.
72. Manaris, B., 2007. Dropping cs enrollments: Or the emperor's new clothes? ACM SIGCSE Bulletin 39, 6–10. doi:10.1145/1345375.1345377.
73. Marin-Garcia, J.A., Mauri, J.L., 2007. Teamwork with university engineering students. group process assessment tool, in: The 3rd WSEAS/IASME International Conference on Educational Technologies, p. 391–396.
74. Mathrani, A., C.S., Ponder-Sutton, A., 2016. Dropping cs enrollments: Or the emperor's new clothes? Educational Technology Society 19, 5–17.
75. Mcdowell, C., Werner, L., Bullock, H.E., Fernald, J., 2006. Pair programming improves student retention, confidence, and program quality. Communications of the ACM 49, 90–95. doi:10.1145/1145287.1145293.
76. Michael, D., Chen, S., 2005. How college affects students: A third decade of research. John Wiley Sons, Inc.
77. Michael, D., Chen, S., 2006. Serious Games: That Educate, Train, and Info. Thomson Course Technology.
78. Miljanovic, M.A., Bradbury, J.S., 2017. Robobug: A serious game for learning debugging techniques, in: The 2017 ACM Conference on International Computing Education Research - ICER 17. doi:10.1145/3105726.3106173.
79. Navimipour, N.J., Zareie, B., 2015. A model for assessing the impact of e-learning systems on employees' satisfaction. Computers in Human Behavior 53, 475–485. doi:10.1016/j.chb.2015.07.026.
80. Olipas, C., 2022. A phenomenological study on the feelings, challenges, and difficulties experienced by information

technology students in learning computer programming. Path of Science 8, 2001–2006. doi:10.22178/pos.83-3.

81. Omar, N.D., Hassan, H., Atan, H., 2012. Student engagement in online learning: Learners attitude toward e-mentoring, in: Procedia - Social and Behavioral Sciences, p. 464–475. doi:10.1016/j.sbspro.2012.11.351.
82. Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., Lahmine, S., 2015. Learning basic programming concepts by creating games with scratch programming environment. Procedia - Social and Behavioral Sciences 191, 1479–1482. doi:10.1016/j.sbspro.2015.04.224.
83. Papadopoulos, Y., Tegos, S., 2012. Using microworlds to introduce programming to novices, in: The 16th Panhellenic Conference on Informatics. doi:10.1109/pci.2012.18.
84. Phuong, D. D., H.F.T.H., Shimakawa, H., 2008. Collaborative learning environment with convincing opinions for novice programmers, in: The 5th WSEAS/IASME International Conference on Engineering Education, pp. 88–94. doi:10.1109/pci.2012.18.
85. Piwek, P., Savage, S., 2020. Challenges with learning to program and problem solve: An analysis of student online discussions, in: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, Association for Computing Machinery, New York, NY, USA. p. 494–499. URL: https://doi.org/10.1145/3328778.3366838, doi:10.1145/3328778.3366838.
86. Porter, L., Guzdial, M., Mcdowell, C., Simon, B., 2013. Success in introductory programming: What works? Communications of the ACM 56, 34–36. doi:10.1145/2492007.2492020.
87. Porter, L., Lee, C., Simon, B., Guzdial, M., 2017. Preparing tomorrows faculty to address challenges in teaching computer science. Communications of the ACM 60, 25–27. doi:10.1145/3068791.
88. Qian, Y., Lehman, J., 2017. Students' misconceptions and other difficulties in introductory programming: A literature review. ACM Trans. Comput. Educ. 18. URL: https://doi.org/10.1145/3077618, doi:10.1145/3077618.
89. Queirós, R., Pinto, M., Terroso, T., 2020. Computer Programming Education in Portuguese Universities, in: Queirós, R., Portela, F., Pinto, M., Simões, A. (Eds.), First International Computer Programming Education Conference (ICPEC 2020), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany. pp. 21:1–21:11. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12308, doi:10.4230/OASIcs.ICPEC.2020.21.
90. Ragonis, N., Ben-Ari, M., 2005. A long-term investigation of the comprehension of oop concepts by novices. Computer Science Education 15, 203–221. doi:10.1080/08993400500224310.
91. Rajeev, S., Sharma, S., 2018. Educational game-theme based instructional module for teaching introductory programming, in: IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society. doi:10.1109/iecon.2018.8592835.
92. Ramabu, T.J., Sanders, I., Schoeman, M., 2021. Teaching and learning cs1 with an assist of manipulatives, in: 2021 IST-Africa Conference (IST-Africa), pp. 1–8.
93. Roberts, J., Styron, R., 2011. Student satisfaction and persistence: Factors vital

to student retention. Research in Higher Education Journal 6, 1–18.
94. Robins, A., Rountree, J., Rountree, N., 2003. Learning and teaching programming: A review and discussion. Computer Science Education 13, 137–172. doi:10.1076/csed.13.2.137.14200.
95. Rosenberg, J., Kölling, M., 1997. Testing object-oriented programs: Making it simple. ACM SIGCSE Bulletin 29, 77–81. doi:10.1145/268085.268115.
96. Ross, M.J., 2002. Guiding students through programming puzzles: Value and examples of java game assignments. ACM SIGCSE Bulletin 34, 94–98. doi:10.1145/820127.820175.
97. Rozali, N.F., Zaid, N.M., 2017. Code puzzle: Actionscript 2.0 learning application based on problem based learning approach, in: 2017 6th ICT International Student Project Conference (ICT-ISPC). doi:10.1109/ict-ispc.2017.8075329.
98. Rubin, B., Fernandes, R., Avgerinou, M.D., 2013. The effects of technology on the community of inquiry and satisfaction with online courses. The Internet and Higher Education 17, 48–57. doi:10.1016/j.iheduc.2012.09.006.
99. Savage, S., Piwek, P., 2019. Full report on challenges with learning to program and problem solve: an analysis of first year undergraduate open university distance learning students' online discussions.
100. Seyal, A.H., Mey, Y.S., Matusin, M.H., Siau, H.N.H., Rahman, A.A., 2015. Understanding students learning style and their performance in computer programming course: Evidence from bruneian technical institution of higher learning. International Journal of Computer Theory and Engineering 7, 241–247. doi:10.7763/ijcte.2015.v7.964.
101. Simon, Luxton-Reilly, A., Ajanovski, V.V., Fouh, E., Gonsalvez, C., Leinonen, J., Parkinson, J., Poole, M., Thota, N., 2019. Passrates in introductory programming and in other stem disciplines, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/3344429.3372502, doi:10.1145/3344429.3372502.
102. Singh, K.Y., 2006. Fundamental of research methodology and statistics. New Age International.
103. Sithole, A., Chiyaka, E.T., McCarthy, P., Mupinga, D.M., Bucklein, B.K., Kibirige, J.S., 2017. Student attraction, persistence and retention in stem programs: Successes and continuing challenges. Higher Education Studies 7, 46–59.
104. Sloan, R.H., Troy, P., 2008. Cs 0.5: A better approach to introductory computer science for majors. ACM SIGCSE Bulletin 40, 271–275. doi:10.1145/1352322.1352230.
105. Smith, P., 2013. Serious Games 101. New Age International.
106. Sprint, G., Cook, D., 2015. Enhancing the cs1 student experience with gamification, in: 2015 IEEE Integrated STEM Education Conference. doi:10.1109/isecon.2015.7119953.
107. Sullivan, G.M., Feinn, R., 2012. Using effect size—or why thepvalue is not enough. Journal of Graduate Medical Education 4, 279–282. doi:10.4300/jgme-d-12-00156.1.
108. Sun, P.C., Tsai, R.J., Finger, G., Chen, Y.Y., Yeh, D., 2008. What drives a successful e-learning? an empirical

investigation of the critical factors influencing learner satisfaction. Computers Education 50, 1183–1202. doi:10.1016/j.compedu.2006.11.007.

109. Swamidurai, R., Kannan, U., 2016. A preliminary report on improving student motivation and persistence in computer programming courses with software inspection, in: ASEE Gulf-Southwest Annual Conference, TCU. doi:10.18260/1-2-620-38953.
110. Talebian, S., Mohammadi, H.M., Rezvanfar, A., 2014. Information and communication technology (ict) in higher education: Advantages, disadvantages, conveniences and limitations of applying e-learning to agricultural students in iran. Procedia - Social and Behavioral Sciences 152, 300–305. doi:10.1016/j.sbspro.2014.09.199.
111. Talton, J.O., Peterson, D.L., Kamin, S., Israel, D., Al-Muhtadi, J., 2006. Scavenger hunt: Computer science retention through orientation. ACM SIGCSE Bulletin 38, 443–447. doi:10.1145/1124706.1121478.
112. Teo, T., 2014. Preservice teachers' satisfaction with e-learning. Social Behavior and Personality: An International Journal 42, 3–6. doi:10.2224/ sbp.2014.42.1.3. Tobias, S., Fletcher, J.D., 2007. What research has to say about designing computer games for learning. Educational Technology 47, 20–29.
113. Topalli, D., Cagiltay, N.E., 2018. Improving programming skills in engineering education through problem-based game projects with scratch. Computers Education 120, 64–74. doi:10.1016/j.compedu.2018.01.011.
114. Triplett, D., 2002. An interview with robocode creator mat nelson. Developer Works, IBM's Resource for Developers.
115. Tîrziu, M.A., Vrabie, C., 2015. Education 2.0: E-learning methods. Procedia - Social and Behavioral Sciences 186, 376–380. doi:10.1016/j. sbspro.2015.04.213. United-Nations, 2018. World Economic Situation and Prospects. United Nations.
116. Vogel, J., V.D.S.C.B.J.B.C.A.M.K., Wright, M., 2006. Computer gaming and interactive simulations for learning: A meta-analysis. Journal of Educational Computing Research 34, 229–243.
117. Wang, T.H., 2014. Developing an assessment-centered e-learning system for improving student learning effectiveness. Computers Education 73, 189–203. doi:10.1016/j.compedu.2013.12.002.
118. Watson, C., Li, F.W., 2014. Failure rates in introductory programming revisited, in: The 2014 conference on Innovation technology in computer science education - ITiCSE 14, pp. 39–44. doi:10.1145/2591708.2591749.
119. Watson, S., Lipford, H.R., 2019. Motivating students beyond course requirements with a serious game, in: Tthe 50th ACM Technical Symposium on Computer Science Education - SIGCSE 19, pp. 211–217. doi:10.1145/3287324.3287364.
120. Weaver, K.F., Morales, V.C., Dunn, S.L., Godde, K., Weaver, P.F., 2017. An Introduction to Statistical Analysis in Research: With Applications in the Biological and Life Sciences. John Wiley Sons, Inc.
121. Wilson, B.C., 2002. A study of factors promoting success in computer science including gender differences.

Computer Science Education 12, 141–164. doi:10.1076/csed.12.1.141.8211.

122. Wolz, U., Barnes, T., Parberry, I., Wick, M., 2006. Digital gaming as a vehicle for learning. ACM SIGCSE Bulletin 38, 394–395. doi:10.1145/1124706.1121463.
Woodfield, R., 2014. Undergraduate retention and attainment across the disciplines. York: Higher Education Academy.

123. Xu, D., Huang, W.W., Wang, H., Heales, J., 2014. Enhancing e-learning effectiveness using an intelligent agent-supported personalized virtual learning environment: An empirical investigation. Information Management 51, 430–440. doi:10.1016/j.im.2014.02.009.

124. Yan, L., 2009. Teaching object-oriented programming with games, in: 2009 Sixth International Conference on Information Technology: New Generations. doi:10.1109/itng.2009.13.

125. Zareie, B., Navimipour, J.N., 2016. The effect of electronic learning systems on the employee's commitment. International Journal of Management Education 14, 167–175. doi:10.1016/j.ijme.2016.04.003.

126. Zhang, J., Caldwell, E.R., Smith, E., 2013a. Learning the concept of java inheritance in a game, in: The 18th International Conference on Computer Games (CGAMES), pp. 212–216. doi:10.1109/cgames.2013.6632635.

127. Zhang, X., Zhang, C., Stafford, T., Zhang, P., 2013b. Teaching introductory programming to is students: The impact of teaching approaches on learning performance. Journal of Information Systems Education 24, 147–155.

128. Zhao, D., Chis, A., Muntean, G.M., Muntean, C.H., 2018. A large-scale pilot study on game-based learning and blended learning methodologies in undergraduate programming courses, in: The 10th International Conference on Education and New Learning Technologies. doi:10.21125/edulearn.2018.0948.

129. Zhao, D., Muntean, C.H., Chis, A.E., Muntean, G.M., 2021. Learner attitude, educational background, and gender influence on knowledge gain in a serious games-enhanced programming course. IEEE Transactions on Education 64, 308–316. doi:10.1109/TE.2020.3044174.

130. Zhao, D., M.C.H., Muntean, G., 2019. The restaurant game: a newton project serious game for c programming courses, in: The Society for Information Technology Teacher Education International Conference, pp. 2121–2128.

131. Zweben, S., 2008. 2006-2007 taulbee survey. Computing Research News 3.

132. Zweben, S., 2009. 2007-2008 taulbee survey: Upward trend in undergraduate cs enrollment; doctoral production continues at peak levels. Computing Research News.

133. Zweben, S., 2010. 2008-2009 taulbee survey undergraduate cs enrollment continues rising; doctoral production drops. Computing Research News.

134. Zweben, S., Bizot, B., 2017. 2017 cra taulbee survey another year of record undergrad enrollment; doctoral degree production steady while master's production rises again. Computing Research Association.