Computers & Education 75 (2014) 72–81

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Cognitive outcomes from the Game-Design and Learning (GDL) after-school program

Mete Akcaoglu [a,*], Matthew J. Koehler [b,1]

[a] West Virginia University, College of Education and Human Services, 355 Oakland St., 506B Allen Hall, Morgantown, WV 26505, USA
[b] Michigan State University, Department of Counseling, Educational Psychology, and Special Education, 600 Farm Lane, Room 509, East Lansing, MI 48824, USA

ARTICLE INFO

Article history:
Received 5 December 2013
Received in revised form
5 February 2014
Accepted 6 February 2014

Keywords:
Game-design
Problem-solving
Quasi-experimental
Constructionism

ABSTRACT

The Game-Design and Learning (GDL) initiative engages middle school students in the process of game-design in a variety of in-school, after-school, and summer camp settings. The goal of the GDL initiative is to leverage students' interests in games and design to foster their problem-solving and critical reasoning skills. The present study examines the effectiveness of an after-school version of the GDL program using a quasi-experimental design. Students enrolled in the GDL program were guided in the process of designing games aimed at solving problems. Compared to students in a control group who did not attend the program ($n = 24$), the children who attended the GDL program ($n = 20$) showed a significant increase in their problem-solving skills. The results provide empirical support for the hypothesis that participation in the GDL program leads to measurable cognitive changes in children's problem-solving skills. This study bears important implications for educators and theory.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Despite the widely-recognized importance of problem-solving as a higher-order thinking skill, during formal schooling students do not get many chances to solve complex problems that they face in their daily lives (Jonassen, 2004; Mayer & Wittrock, 2006). One reason for this is that schools place a heavy emphasis on covering and delivering content knowledge. In addition, schools frequently introduce problem-solving to students in the form of repetitive and well-structured problems (Perkins, 1986). Practice in well-structured problems (i.e., problems with one possible solution) do not help students in gaining skills to solve real-life problems, which are characterized as ill-structured, complex, and having usually more than one solution (Jonassen, 2000).

Because of the shortcomings of formal education in teaching problem-solving skills, researchers and educators sought alternative ways to teach students skills necessary to solve complex problems (Mayer & Wittrock, 1996). One such alternative has been using programming to teach problem-solving. Pioneered by Seymour Papert (1980), programming, and later software and game-design, has received an important amount of attention both from researchers and educators. Papert (1980) argued that through programming children get opportunities to "tinker" with objects, and in the process learn about their own thinking (Guzdial, 2004).

Recently, using game-design as a context to teach higher-order thinking skills has received interest from researchers (e.g., Hwang, Hung, & Chen, 2013; Ke, 2014; Papastergiou, 2009), but empirical support for the cognitive benefits of these tasks has been slow to emerge (Denner, Werner, & Ortiz, 2012). In this paper, building upon our previous work (Author, 2012), we aim to provide empirical support for the impacts of learning game-design on students' problem-solving skills, especially in system analysis and design, decision-making, and troubleshooting.

* Corresponding author. Tel.: +1 304 293 4075.
E-mail addresses: meakcaoglu@mail.wvu.edu, mete.akca@gmail.com (M. Akcaoglu), mkoehler@msu.edu (M.J. Koehler).
[1] Tel.: +1 517 353 9287; fax: +1 517 353 6393.

http://dx.doi.org/10.1016/j.compedu.2014.02.003
0360-1315/© 2014 Elsevier Ltd. All rights reserved.

2. Background

2.1. Problem-solving

Problem-solving is reaching a desired goal state from a given state by overcoming barriers in between and when there are no obvious ways of reaching the goal state (Mayer & Wittrock, 2006). By this definition, *problem* refers to situations when a problem solver has a goal but does not have obvious methods to get there. The act of design, where the designer's goal is to reach a desired goal state (i.e., the final design) can be considered as an example of a problem situation (Smith & Boling, 2009).

The process of problem-solving involves activation of important cognitive skills, and therefore, problem-solving is often seen synonymous to *thinking* (Mayer, 1977). From a cognitive perspective, the problem-solving process involves representing, planning, executing, and evaluating (Jonassen, 2004; Polya, 1957). Representing refers to converting the external representation of a problem into an internal mental representation. This can also be equated with *understanding* the problem. Planning involves devising a solution, specifically by breaking the problem into its components. Executing is putting the plan into action, and finally, evaluating refers to checking to see if the executed plan actually helped get to the desired goal state.

2.2. Methods of teaching problem-solving

Researchers (e.g., Mayer & Wittrock, 1996, 2006) have identified methods that have proven to be effective when teaching problem-solving. Some of these instructional methods can also lead to meaningful learning by helping students "to apply what they have learned to solving new problems" (Mayer & Wittrock, 2006, p. 290).

Reviews by Mayer and Wittrock (1996, 2006) provided researchers and educators with a list of effective methods of teaching problem-solving. One of the methods of teaching problem solving is *teaching basic skills*. According to this method, students are taught low-level skills in order to give them more cognitive power to execute higher-level skills when tackling more complex problems (e.g., load-reduction methods) (Mayer & Wittrock, 2006). Another method is *teaching for understanding*, which involves techniques like *generative methods* where learners are led to think about the relationship between their existing and new knowledge. Teaching by using a*nalogies* is another effective method to teach students problem-solving. In teaching by analogies, the aim is to show learners how problems that differ on the surface can have structural similarities and can be solved using the same methods (Gick & Holyoak, 1980). Finally, *teaching (metacognitive) thinking skills* is another method of problem solving, where the aim is openly showing the students important metacognitive skills that they can use to solve problems. As it will be described later, during the GDL program, these methods were incorporated at varying degrees in order to teach students problem-solving skills.

2.3. Problem types

Although by definition problems bear underlying structural similarities, there are different types of problems in terms of the cognitive processes they require (Jonassen, 2000). Three specific types of problems that we specifically chose to investigate in this research were system analysis and design, decision-making, and troubleshooting. These problem types were chosen because they are "widely applicable and occur in a variety of [real-life] settings" (OECD, 2003, p. 2).

System analysis and design involves understanding the connection between parts of a system, and being able to create complex systems by bringing together a number of interdependent variables to make a single, harmonious, and functional unit (Jonassen, 2000). In our daily lives, we are faced with situations that ask us to analyze and/or design systems. For example, when people visit a new country or a city, a common system analysis and design task is to understand the transportation system and figuring out the best, safest and cheapest ways to travel around. Similarly, if one tries to learn a new language, the analysis of the target language can be categorized as a system analysis task (i.e., figuring out the grammatical rules), while forming of new sentences would be "design," because sentences can be formed in an infinite number of ways (Chomsky, 1959). Finally, another common context where people make use of system analysis and design skills is science jobs. For example, in order to identify an environmental or a medical problem, scientists analyze data, and look for patterns to understand the "system" behind these problems. Solution to these problems, then, needs to be "designed," because they are novel and unique, and the existing solutions do not work. For example, to help Steve Jobs fight against his pancreas cancer, a team of scientists designed and developed special medication that targeted slowing the growth of the cancerous cells in his body by analyzing the patient's DNA structure (Isaacson, 2011).

Decision-making involves solving a dilemma (Jonassen, 2000). Solving a dilemma is a complex task, because dilemmas can be ill-structured and "often there is no solution that is satisfying or acceptable to most people, and there are compromises implicit in every solution" (p. 80). Similar to system analysis and design, decision-making problems are commonly found in our daily lives. Our daily lives are filled with moments where we are required to make decisions, such as:

> "Should I move in order to take another job; which school should my daughter attend; which benefits package should I select; which strategy is appropriate for a chess board configuration; how am I going to pay this bill; what's the best way to get to the interstate during rush hour; how long should my story be."
>
> Jonassen, 2000, p. 76

For this reason, becoming mindful about the importance of making informed decisions, analyzing the cost-effectiveness, and predicting the future impacts of these decisions are important skills for young children to develop.

Finally, *troubleshooting* requires identifying and fixing an inoperable component of an otherwise working system or a mechanism (Jonassen, 2000). Similar to system analysis and design, troubleshooting requires an understanding of the system, but in troubleshooting the focus is on identifying the element(s) that brings a system to a halt. Figuring out why certain software or an electronics device is

misbehaving, for example, and then taking necessary steps to get it working again is a common troubleshooting scenario people face daily (National Research Council, 1999). Our daily lives are filled with situations where we are required to troubleshoot a non-working system. For example, when someone realizes that her car is not working, the process of troubleshooting starts. To identify the source of the problem, the person takes some steps: check the battery, check the gas level, etc. Each of these sources requires certain tests to be confirmed as the actual source. For example, realizing the lights of the car are working indicates that the problem might not be due to the battery, and helps the person rule out an alternative. For success in future careers, being good troubleshooters is important. For instance, a key skill in computer programming is debugging, or finding the error (also known as "bugs") in code to get it to function. Without doubt, almost every software application that people use today goes through extensive stages of debugging. Therefore, for future programmers, understanding the process of troubleshooting and improving their skills in this area is important.

All of these three problem types are complex in nature, and cover the important problem-solving processes identified within the problem-solving domain (OECD, 2003). Therefore, providing young students with practices in tackling these problems helps them develop the ability to solve complex problems. As mentioned before, especially in today's increasingly technology-filled world, all of these problems are encountered frequently in our daily lives. Understanding the systems around us, designing new systems, troubleshooting them, and making informed decisions based on data are important skills to possess for the young children to be successful in their future lives and careers.

2.4. Research on using computers to teach thinking skills

Starting in the 1970s, researchers in the field of educational technology became interested in how technologies, especially programming with LOGO, could foster children's thinking skills. During the 1980s and 1990s, it was a hot debate whether learning programming, mainly through discovery learning methods, had any effects on students' general problem-solving skills (Mayer & Wittrock, 1996). Pioneering the field, Papert (1980) argued that by programming with LOGO (a simplified programming language), children showed improvements in their thinking skills.

In an early research, Choi and Repman (1993) found that learning to program with Pascal led to improvements in college students' problem-solving skills. Similarly, comparing a group of high-school students who were learning to program to a group of students who were learning basic computer literacy, Reed and Palumbo (1988) found that the group who participated in a 15-week BASIC instruction ($n = 11$) showed a significant increase in their problem-solving ability compared to their peers ($n = 11$) who received computer literacy instruction during the same period. Finally, in a recent study by Unuakhalu (2009), 40 undergraduate students were divided into two groups to receive instruction on programming only, or programming and tasks relating it to everyday problems. At the end of the 8-week treatment, students who received instruction on programming embedded in everyday tasks performed better on problem-solving posttest, leading authors to conclude that embedding programming instruction into everyday tasks was an effective method.

Based on the early studies investigating the impacts of programming on thinking skills, recently researchers argued that learning game-design would lead to similar improvement of thinking skills. Research to support these claims, however, has been lacking, or anecdotal at best. For example, in their recent research, Richards and Wu (2012) and Wu and Richards (2011) explored whether game-design activities were effective in teaching computational thinking skills. Although the researchers collected data from several sources (reflective short answer questionnaires, field notes, student game-design journals, and semi-structured interviews), the results were based on single cases or anecdotes rather than strong empirical data. Similarly, in another recent study Games and Kane (2011) looked at the relationship between game-design activities and the students' thinking skills and STEM knowledge. In this study, as well, the authors fell short in providing empirical support for their claims.

Recently, researchers (e.g., Baytak & Land, 2010; Denner et al., 2012) investigated the impacts of learning game-design on students' content knowledge, or programming skills, and provided us with support for the positive impacts of learning game-design on students' content knowledge (e.g., Baytak & Land, 2010), and programming skills (e.g., Denner et al., 2012). Similarly, other recent research indicated that through learning game-design students showed improvements in their attitudes toward learning (Hwang et al., 2013) and math (Ke, 2014), while the process was also believed to positively impact students' computational thinking skills (Denner et al., 2012; Ke, 2014).

Finally, the results of a study conducted previously by Author (2012) showed that following a curriculum that aimed to teach problem-solving through game-design and programming activities led to improvements in students' problem-solving skills. In this, the authors found that the students who attended a 40-h summer game-design program ($n = 18$) showed significant improvements in their problem-solving skills. This work, however, had some methodological limitations, such as not having a control group to compare the effects to.

In the current study, we extend the efforts of the previous research, as well as our own, to understand if the process of learning game-design can also be a good context to teach problem-solving skills. Early work enlightened us as to the positive impacts of learning game-design on children's programming, math, or computational thinking skills. Game-design tasks have also received support for their positive impact on children's motivation to learn or solve problems (e.g., Hwang et al., 2013). We, however, do not know if students can also find practice and improve their skills in problem-solving when they are engaged in game-design.

3. The study context: the GDL after-school program

In an effort to bring the benefits and the affordances of game-design and programming to middle school students, we developed an initiative called The Game-Design and Learning program (or GDL for short). Since its inception, the GDL initiative has engaged middle school students in the process of game-design in a variety of in-school, after-school, and summer camp settings. The goal of the initiative is to leverage students' interests in games and design to foster their problem solving and critical reasoning skills.

The after-school format of the GDL program was developed as a series of 3-h sessions to be offered five times during weekends (for a total of 15 h). The goals of the GDL program were to: (a) teach students how to design digital games; (b) provide students with a broader understanding (and skills) in programming; (c) give students practice in becoming producers of digital media rather than consumers; and (d) teach students problem-solving skills by using game-design as the context for ill-structured problems.

When designing the GDL curriculum, we paid specific attention to incorporating well-known instructional methods for solving problems, as teaching problem-solving was one of the main goals. In the next section, we summarize the overall structure of the GDL curriculum, as well as some of the instructional methods used, in order to give the readers a feel of the instructional context of the GDL program. Readers interested in a broader discussion of the theory behind the design of the curriculum, or a more detailed account of instruction can find such detailed reports elsewhere (Akcaoglu, 2014).

3.1. Pedagogical principles

Two main pedagogical approaches that guided the design and delivery of the GDL program were *constructionism* and *guided discovery learning*. Constructionism is a derivative of constructivism – the idea that learning is active process of constructing knowledge (Greeno, Collins, & Resnick, 1996). Replacing "v" with an "n," Papert (Papert & Harel, 1991) argued that learning is much more meaningful when the construction process leads to the creation of socially meaningful artifacts (Ackermann, 2001). Built upon the idea of constructionism, during the GDL program, the students were encouraged to construct their own knowledge of game-design and produce socially meaningful and engaging artifacts: games (Hwang et al., 2013; Li, 2010).

During the process of construction and design of their games, learners were also encouraged to freely work and *discover* knowledge (Mayer, 2004). During the GDL program, guided discovery approach was chosen over pure discovery, and learners were provided with minimal, but sufficient, guidance and feedback during the game-design process. The reason for this choice was because pure discovery received criticism (Kirschner, Sweller, & Clark, 2006; Mayer, 2004) for being taxing on learners' cognition (i.e., learners are expected to discover the connections between new and existing knowledge on their own). As opposed to pure discovery, in guided discovery, the learning process is support by minimal guidance from knowledgeable others. This support helps alleviate the cognitive burden on the learners and benefits the learners more, compared to pure discovery (Kirschner et al., 2006; Mayer, 2004).

3.2. GDL Curriculum and activities

Our curricular goals included teaching students game-design, problem-solving, and programming skills. Although the goals of teaching game-design and programming were natural outcomes of participating in the GDL program (or any digital game-design program for that matter), in order to teach students problem-solving, the GDL curriculum and activities were shaped according to the theories of teaching problem solving. Specifically, the activities were designed to utilize four main methods of teaching problem solving: teach problem-solving skills directly, teach for understanding, teach by using analogies, and teach (metacognitive) thinking skills (Mayer & Wittrock, 1996, 2006). Based on these principles and curricular goals, four main types of activities were offered during the GDL program: game-design, problem solving, troubleshooting, and free-design. As it can be seen in Fig. 1, game-design activities were the through line during GDL program, and other activities (e.g., problem-solving) were introduced later on as students gained more experience and confidence in game-design and programming.

Main goal of *game-design activities* was to teach students the basics of the game-design process. During these activities students also learned background knowledge and skills in using the game-design software (Microsoft Kodu) and computer programming that are necessary for designing, programming and creating digital games. This was based on the idea that learning the basic skills (i.e., teaching basic skills) of game-design and programming would lower the burden on students' cognition and help them better focus on later activities that involved solving complex problems.

During game-design activities, the students were guided in creating games (from simple to more complex), through a series of instructor-led sessions that were followed by free-design sessions where students improved upon their designs and explored concepts in game-design and programming. Also, during game-design activities, the students got hands-on experiences in (a) system analysis, by seeing how games are complex systems, made up of many interrelated variables; and (b) game-design, by creating their own games, which are complex systems. This was especially emphasized when students were given flowcharts of games that they needed to create, and were asked to create flowcharts for the games that they planned to create.

Problem-solving activities were introduced only after the students were comfortable and competent using Kodu and were familiar with the game-design process. In other words, by the time the problem-solving activities were introduced, the students had already created at least three games with varying degrees of complexity, become familiar with Kodu, and practiced the basic game-design skills multiple times. The purpose of these activities was to give students the opportunities to pair the process of *problem solving* with *design*. To this end, problem-solving activities included reading a problem scenario (e.g., a story built on predator-prey relationship), solving the problem by interpreting the data and the relationship among the variables presented in the scenario, and then creating a replica of the scenario by creating a simulation of it in Kodu.

During problem-solving activities, instructor guidance was available in the form of directing students through the steps of solving a problem. The guidance was provided in two specific forms: (a) initial guidance given to the entire class in solving the problem scenario, and (b) one-on-one guidance to individual students when they struggled with the programming or the design of the simulation. For example, in the first problem-solving activities, the instructor presented the problem scenario, as well as elicited solutions from the entire class. During

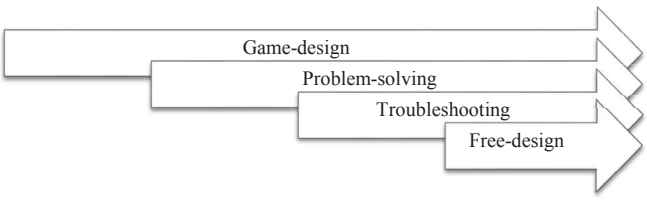

Fig. 1. Progression of different activities during GDL courses.

this process, the instructor guided the students in following the steps to solve problems (e.g., understand the problem, and plan a solution). In fact, through this open guidance, another method of teaching problem solving, teaching (meta-cognitive) skills directly (i.e., identify elements, plan solution, execute plan, evaluate outcome), was incorporated.

Other effective methods of teaching problem-solving were also used within problem-solving activities. For example, by using analogies, the students were introduced to problem scenarios built on the same underlying structures but differing on the surface. This way, the students were given chances to work with analogous problems and identify patterns while solving problems. Finally, the idea to use Kodu to recreate the problem scenarios was based on the idea that students learn better when they create their own personal representations of the problems, connecting old and new knowledge (i.e., generative methods of teaching problem solving).

In addition to activities purely focusing on game-design and problem solving, during GDL students were also offered activities where the main task was to fix a broken game: *troubleshooting*. By analyzing the structure of a given game, during the troubleshooting activities, the students identified the relationship between its elements, and made necessary changes to make the game working again. It should be added that although not explicitly integrated, troubleshooting is a natural part of the game-design process. Therefore, the students found many opportunities to troubleshoot their games during game-design, problem-solving, or free-design activities.

Finally, during *free-design activities* the students were asked to create a game of their own choosing. These free-design activities were purposefully offered toward the end of the GDL program, only after the students gained enough skill and confidence in game-design, programming, and problem solving, before taking on such an open-ended task like free-design. These activities provided students with chances to create games that they personally valued. Providing students with free-design activities toward the end was also important because it both involved elements from previous activities (game-design, problem-solving, troubleshooting) and contextualized these skills in a meaningful and personal context such as creating their own games.

During free-design activities, as in the other activities, the students also got chances to practice their decision-making skills. It was a common occurrence for students to decide how to balance the amount of objects in a game based on the capacity of the computers to run their games. In other words, they were constantly reminded to be mindful about how much processing power their computers had and how they can make the right decisions where there is minimal compromise. For example, after realizing that his game world was too large for Kodu to run, one student decided to make all the characters in his game smaller (by changing their size scale), and removed elements that were not crucial to the gameplay. By this way, he managed to overcome the restriction in Kodu without compromising the integrity of his game.

4. Purpose

The purpose of this study was to provide empirical support for the positive impacts of learning game-design on students' problem solving skills. In other words, we aimed to investigate whether students improved in their problem-solving skills by attending the GDL program where they got opportunities to practice their problem-solving skills through game-design activities.

In our previous research (Akcaoglu, Boyer, & Kereluik, 2012), we found initial evidence of the connection between learning game-design and improvement in students' problem-solving skills. This early research was, however, pre-experimental. More specifically, there was not a control group to compare the outcomes of the GDL courses. This especially limited interpretation of our results due to possible testing and maturation effects. In other words, the improvement the students showed from pre to the post test was confounded by the fact that the students took the same test twice, and we did not have another way to eliminate this rival hypothesis. In our current study, we sought to overcome this methodological weakness inherent to our previous work by comparing outcomes of the students who attended the GDL program to a control group. Therefore, our research question was:

- Are there differences between control and GDL students in terms of their gains in problem solving skills?

5. Method

5.1. Participants

A total of 44 students participated in the current study: 20 students in GDL condition and 24 in the control condition. Data for the GDL group came from two groups of students who attended the GDL programs that were offered during fall of 2012 in Istanbul, Turkey ($n = 13$) and Michigan, USA ($n = 7$). The age average of the students in the GDL group was 11.9, ranging between 11 and 13. Four participants were female (1 female in the US group and 3 females in the Turkey group), and 16 were male. The students in both Turkey and the US sites came from upper-middle class families.

As for the demographics of the control group, a total of 24 (12 male and 12 female) students with an age average of 13.4, ranging from 12 to 14, were recruited as the control group. These students were also attending the school where the GDL program was offered in Turkey. Similar to the experimental groups, students in the control group were also coming from upper-middle class families. In order to minimize selection biases, the students in the control group were selected from students who could not attend the fall GDL classes due to space limitations and were interested in a summer offering. With this recruitment strategy we were able to match our students in terms of their interest in game-design.

Although the activities and scope of the GDL program were kept exactly the same at both sites, since the students were from two different countries, before merging the data, we tested to see if the students at the two GDL sites were similar in initial levels and showed similar gains. Our analysis indicated that there were not any significant differences between the experimental groups in terms of their initial levels of problem solving, (Wilks's $\Lambda = 0.866$), $F(3, 16) = 0.827$, $p = 0.498$, $\eta^2 = 0.13$; as well as the gains they showed after attending the GDL program, (Wilks's $\Lambda = 0.903$), $F(3, 16) = 0.571$, $p = 0.642$, $\eta^2 = 0.097$. The two GDL groups, therefore, were combined and treated as one group for the further analyses.

5.2. Instruments

Students' problem-solving skills were measured by a publicly available and internationally validated assessment, *The Program for International Student Assessment* (PISA), prepared and offered through *Organisation for Economic Co-operation and Development* (OECD, 2003). This assessment was designed to measure students' skill at solving three problem types: system analysis and design, troubleshooting, and decision-making. The instrument had a total of 19 items, in the form of multiple-choice and open-ended questions. The test can be accessed through OECD's website (OECD, 2013).

The PISA test was selected for various reasons. First, the test's reliability and validity were established through rigorous international field-testing (OECD, 2003). Second, its scoring is based on item-response theory (IRT) procedures. IRT provides remedies for several important shortcomings of classical test theory (CTT) (Hambleton, 1990). First, in CTT, item parameters (e.g., item difficulty) are estimated based on the sample at hand, and are not valid for different samples, whereas in IRT the estimates are not sample-specific and can be applied to different samples. Second, in CTT, information regarding what an examinee might do when confronted with a specific item is not known, whereas in IRT students ability estimates can be calculated because "[ability estimates] is obtained by incorporating information about the items (i.e., their statistics) into the ability estimation process" (Hambleton, 1990, p. 99). Hence, by using IRT, we were able to get individual proficiency estimates that were calculated based on well-established item difficulty parameters obtained through previous data and research. Finally, the PISA test was used in this study due to the inclusion of various problem types that are complex in nature and frequently encountered in real-settings: system analysis and design (8 items), decision-making (7 items), and troubleshooting (5 items).

5.3. Dependent variables

System analysis and design refers to analyzing and creating systems commonly found in real-life settings. Designing or analyzing a system is a complex task and success is dependent upon understanding the intricate relationship (recognizing patterns) among different variables. For example, one of the questions in the PISA test required students to understand the logic of a simple programming language that produced a set of geometrical shapes (i.e., system analysis), which was followed by a question requiring students to write to code to create a certain shape (i.e., system design). Students' system analysis and design skill estimates were computed based on their performance average on eight system analysis and design questions in the test.

Decision-making is a complex problem-solving process, involving selecting the best option from many available others (Funke & Frensch, 1995; Huber, 1995; Jonassen, 2000). During the PISA test, students' skill in making decisions was measured by questions that provided them with constraints and asked them to make the best decision in the given circumstances. For example, in one question students were asked to find the best movie and day/time to see it by looking at information tables regarding day, time, and rating information for movies at a movie theater, as well as availability and preference information of a group of 14 year-old students for a week. Based on students' performance average in the seven decision-making questions in the PISA tests, a skill estimate for this domain was calculated.

Troubleshooting is perhaps the most common problem solving task people face in their daily lives (Jonassen, 2000). Troubleshooting requires problem solvers "to comprehend the main features of a system and to diagnose a faulty, or under-performing, feature of the system or mechanism" (OECD, 2003, p. 168) to bring it back to its original working state. During the PISA test the students were tasked with identifying elements of systems that malfunction, and suggesting ways to fix them. For example, in one troubleshooting question the students were asked to propose a method to identify a faulty gate in the gate system of a waterway network around a garden. With this question students not only had a chance to analyze the relationship among the variables of the given system, but also were asked to troubleshoot the system to understand which element(s) was causing the problem. The mean of students' performance in five troubleshooting questions was taken as their troubleshooting skill estimate.

The PISA test was designed to measure students' skills in system analysis and design, troubleshooting, and decision-making in a generic sense. In other words, the questions were not contextualized within game-design. Therefore, skill at solving these problems was not specific to game-design and could be generalized to other domains.

5.4. Procedures

In both the US and Turkish versions, the format of the program, as well as the lead instructor were the same. At both sites, the students attend a 15-h (five 3-h sessions) program. The sequence and content of the instructional activities were also kept exactly the same. The instructional team at the US site consisted of a lead instructor and two assistant instructors, while at the Turkish site there was only one assistant instructor. The lead instructor was the creator of the GDL initiative, who designed the curriculum and the activities. In addition, he had experience teaching game-design, because he had previously run two GDL programs during the summers of 2011 and 2012. The main role of the assistant instructors was to help with classroom organization and management, and they did not have any instructional roles.

For the GDL group, the testing happened on the first day of the GDL program. The students, first, took the pretest of problem solving. The test was in paper-pencil format and took around 40 min to complete. Following the test, the students completed a survey of technology competence and motivation. After completing the testing stage, the students followed the GDL curriculum for five weeks. On the last GDL session, the students took the same problem-solving assessment and the surveys.

The control group did not attend the GDL program, or any other after-school activities. As mentioned previously, they were chosen among students who were willing to attend a summer program in the future. The students in the control group took the pretest of problem solving at the same time as the GDL students (they were taken to a separate classroom). The test was in paper-pencil format and took about 40 min. Following the test, the students were taken back to their classrooms. Similar to the pretest, they took the posttest of problem solving in a separate classroom, at the same time as the GDL group.

The grading of the PISA tests was done by the first author, who was also the lead instructor of the GDL programs in both locations. The grading was done according to the grading rubric prepared and provided by PISA.

It should be noted here that because the GDL program was offered as a single intervention, identifying how much individual activities impacted the students' problem-solving skills was beyond the scope of this research. Instead, we investigated if the GDL intervention

worked as a whole. Therefore, our results we will not be able to talk to the specific reasons why the two groups differed, but will speak to the outcomes of the GDL program as a whole.

6. Results

To calculate each student's problem-solving competency, IRT analyses were conducted using Xcalibre software (Assessment Systems Corporation, 2012). The calculations were made based on the difficulty levels for each question as reported by OECD (2003) and students' performances in the PISA assessment. Calculated according to IRT procedures, each student's problem solving ability was calculated on a scale ranging from −4 to 4. This scale can roughly be interpreted as a student's probability of answering all the items at a test correctly. For example, a student with an ability level of 0 on a specific question is considered to have 50 percent chance (proficiency) of answering the question correctly.

To answer the research question, the gain difference between control and the GDL group students in three problem-solving skills, a repeated-measures multivariate analysis of variance (RM-MANOVA), having two levels of time (pre vs. post) as within subjects factors, and two levels of group (control vs. experimental) as between subjects factor (i.e., mixed-factorial design) was conducted on the dependent variables. The multivariate omnibus for time was significant (Wilks's $\Lambda = 0.616$), $F\,(3, 40) = 8.328$, $p < 0.001$, $\eta^2 = 0.384$; as well as the omnibus for group, (Wilks's $\Lambda = 0.733$), $F\,(3, 40) = 3.0$, $p = 0.006$, $\eta^2 = 0.267$; and the interaction between time and group, (Wilks's $\Lambda = 0.505$), $F\,(3, 40) = 13.063$, $p < 0.001$, $\eta^2 = 0.495$. As it can be seen in Fig. 2 and Table 1, the results indicate that compared to the control group, the students in the GDL group showed significantly larger gains in the three problem-solving skills. In fact, the control group did not improve in any of the problem-solving skills.

In order to further understand the size and nature of the gains the GDL group showed in each of the problem-solving skills, paired-samples *t*-tests were run on students' system analysis design, decision-making, and troubleshooting skills, comparing their pre and the posttest scores. Running multiple *t*-tests in such a manner, however, increases Type I error, namely erroneously rejecting the null hypothesis. Therefore, Bonferroni correction was applied (Field, 2009) by dividing the critical alpha level ($p = 0.05$) by the number of *t*-tests conducted ($n = 3$). This yielded $p = 0.017$ as the critical alpha *p* value for the analyses of the three problem-solving skills.

The results of the *t*-tests indicated that the GDL group demonstrated significant improvements in all three problem-solving skills (system analysis and design, $t(19) = 4.700$, $p < 0.001$; decision-making, $t(19) = 4.694$, $p < 0.001$; troubleshooting, $t(19) = 3.853$, $p = 0.001$). All the effect sizes were large according to Cohen's criteria for effect size interpretation (1988): system analysis and design, $d = 1.062$; decision-making, $d = 1.05$; troubleshooting $d = 0.87$.

6. Discussion and conclusions

Recently, game-design has received attention from educators and researchers for its potential in teaching computational thinking and programming skills (Baytak & Land, 2010; Denner et al., 2012). Evidence supporting the connection between these skills and learning game-design, however, has been slow to emerge. The results of this study showed that, compared to the control group, the students who participated in the GDL program showed significant increases in their problem-solving skills. Large effect sizes point to sizable amounts of change in students' specific problem-solving skills from the pretest to the posttest. More specifically, the GDL group showed large and significant gains in system analysis and design, decision-making, and troubleshooting skills; which confirmed our hypothesis that learning game-design at the GDL program leads to an increase in students' problem-solving skills. This effect was true across different GDL implementations and sites, pointing to the robustness of the GDL curriculum and activities. In addition, because the PISA assessment was designed to measure students' problem-solving skills in a generic sense, it can be argued that the improvement in the students' skills was generalizable to other contexts, and not specific to game-design.

Our previous research (Author, 2012) provided us with a preliminary support for the link between teaching students game-design and changes in their problem solving skills as a result of attending the GDL program. This early work, however, was pre-experimental, and

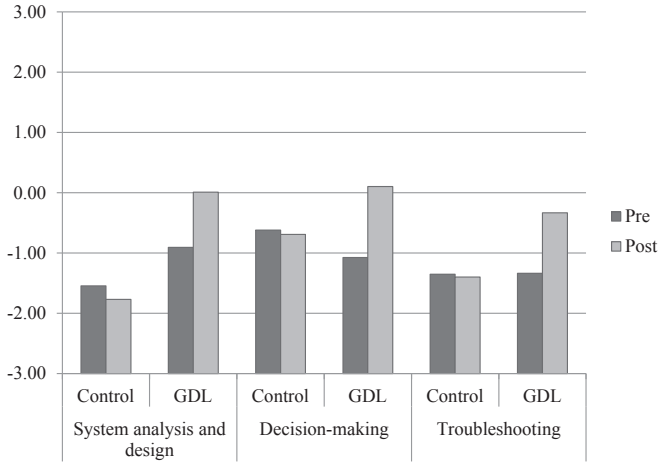

Fig. 2. Pre–Post problem-solving skills for control and GDL groups.

Table 1
Pre–Post test descriptive statistics for GDL and control groups in three problem-solving skills for.

		Control			Experimental		
		M	SD	Gain	*M*	SD	Gain
System Analysis and Design	Pre	−1.54	1.14	−0.23	−0.91	1.32	0.92
	Post	−1.77	1.16		0.01	1.11	
Decision-making	Pre	−0.62	0.62	−0.07	−1.08	1.27	1.18
	Post	−0.69	1.20		0.10	1.14	
Troubleshooting	Pre	−1.35	0.97	−0.05	−1.34	1.11	1.00
	Post	−1.40	1.08		−0.33	1.09	

findings were limited by the shortcomings of such research designs (i.e., maturation and testing). The current research and its findings were necessary next steps to further our previous effort by overcoming some of the shortcomings inherent in our early work.

One of the main reasons why the GDL program helped improve students' problem-solving skills can be the curriculum (and activities) that highlighted the connection between game-design and problem solving. This being said, however, pointing to specific activities that led to improvements in students' skills was not possible to identify with the data at hand. Understanding such specific connections should be investigated in future studies. It can be argued, however, that the activities and the curriculum structure holistically played a key role in the success of the GDL program, and the results should be attributed to the program as a whole.

During GDL, students received many hands-on experiences in *design* tasks. Design problems are complex (i.e., ill-structured) and very open-ended in nature (Bonnardel, 2000), and design process involves satisfying many interrelated variables. In addition, in design problems (as was in the case of game-design) the outcome, or the solution to problems is personally meaningful (Bonnardel & Zenasni, 2010). At GDL, students got first-hand experience in designing, analyzing, and troubleshooting their own games. This can be considered as one of the reasons that led to improvements in their skills to solve complex (design) problems.

An important step in becoming a successful problem solver is to create effective mental representations of the problem and the solution (Jonassen, 2004). As a context for creativity and design, during game-design, students get invaluable chances to create and tinker with their mental representations of problems and solutions in the forms of digital games. According to Bonnardel & Zenasni (2010), creating virtual representations of design problems is a helpful way of understanding and solving them. To this end, they argue that "[creating mental representations] allows designers to reach a better understanding of the design problem and to adopt new points of view about the object to be designed" (p. 182). Engaging students in game-design, while also showing them how design tasks are also complex problems, and finally teaching them skills and metacognitive skills to tackle such complex problems, such as creating visual representations, can be speculated as one of the main reasons behind the effectiveness of GDL program and curricula.

Through the GDL program, problem-solving skill development in three specific problem types was desired: system analysis and design, decision-making, and troubleshooting. Skill at solving these problems is very important for success in real-life settings (Jonassen, 2004). For example, during game-design tasks students were asked to create flowcharts of their games as systems, which can be considered as an important skill to solve system analysis and design type questions. Similarly, in perhaps all GDL activities, the students faced hard-decisions to make. For example, one common decision-making inherent in game-design is deciding how a certain character can be programmed to do a certain task. Identifying the easiest, the lowest resource-intensive, and the most expandable way to program characters in a game can be an important reason for students' development in decision-making skills. Finally, coming across problems is an inherent part of the design process. For example, during GDL, students constantly found themselves struggling to identify characters or snippets of faulty code that prevented their games running in the way they desired. All these processes were inherent parts of the very engaging process of game-design, which eventually resulted in improvements in students' skills.

Although empirical support has been slow to emerge (Denner et al., 2012), in the most recent years, the number of studies looking at the potential of game-design activities as contexts to teach young students certain knowledge and skills, as well as improve their attitudes, have started to increase. The results of this study help to further the field by providing empirical support for the appropriateness of using game-design as meaningful contexts to teach students important skills, such as problem-solving. More specifically, this study adds to the growing body of recent research that indicated that, through game-design, students improved in their math knowledge (Li, 2010, 2013), attitudes toward math (Ke, 2014), programming knowledge (Denner et al., 2012), scientific thinking (Klopfer, Roque, Huang, Wendel, & Scheintaub, 2009), and attitudes toward problem-solving (Hwang et al., 2013). The current work provides support for the connection between learning game-design and improvement in students' actual problem-solving skills in system analysis and design, decision-making, and troubleshooting domains. In addition, in our work we highlight the importance of the role of the theories and pedagogies informing our curriculum, and argue that the careful attention placed on the creation of our curriculum and instructional activities were the main reason for the improvements that we were able to capture in our research. Finally, because our instrument measured students' skills in solving problems in a generic sense, we believe the skills students learned during the GDL program could be applied to other settings. Looking at the possibility of transfer of these skills should be investigated in future work.

7. Limitations of study

As for applicability and generalization of the results to other populations, we suggest that educators and researchers should employ caution when interpreting the results. In this study, we eliminated an important rival hypothesis, the effect of testing and maturation by including a control condition. There were, however, selection issues that could not be overcome in such quasi-experimental designs, and therefore, a true experiment might still be needed. For example, the students in the GDL program, and in the control group, were specific groups of students who showed self-initiated interest in an after-school game-design program. Although we took some measures to prevent

selection issues in this research (e.g., control group included students with similar interest in game-design), the students in this study might bear characteristics (cognitive and motivational) that separate them from the rest of the population, and therefore, confound the findings.

In addition, because our control group did not attend an alternative after-school activity, we cannot isolate the effects of merely participating in after-school activities. Therefore, we do not know how much of the improvement the GDL students showed can be attributed to the GDL program, and how much is due to pure maturation as a result of participating in an after-school program.

Finally, all of our participants in the control group were from the GDL location in Turkey, although our GDL students and data came from locations in Turkey as well as the US. Therefore, this limitation should be considered carefully while interpreting the results of this study.

8. Implications and future study

In addition to leading to significant improvements in students' problem-solving skills, efforts like GDL are important in identifying effective ways to integrate technology into educational contexts. We believe GDL program has potential to offer development in domains other than problem solving, such as important topics from science, technology, education, and math (STEM). For example, through real-life scenarios in human anatomy (e.g., how diseases spread throughout the body), the students can be asked to identify the elements of the scenario, and then asked to recreate this in the chosen game-design software. Through such engaging tasks students' knowledge and skill in STEM subjects can be improved, as well as their interest in these subjects and future STEM careers.

References

Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: what's the difference? *Future of Learning Group Publication, 5*(3), 1–11. DOI:10.1.1.132.4253.

Assessment Systems Corporation. (2012). *xCalibre 4.1.6.1*. Retrieved from http://www.assess.com/xcart/product.php?productid=569.

Akcaoglu, M., Boyer, D. M., & Kereluik, K. (2012). Teaching problem solving through game design: Reflections on Game Design and Learning (GDL) summer camp. In P. Resta (Ed.), *Proceedings of society for information technology and teacher education international conference* (pp. 3–7). VA: AACE: Chesapeake. Retrieved from http://www.editlib.org/p/39531.

Akcaoglu, M. (2014). *Teaching problem solving through making games: Design and implementation of an innovative and technology-rich intervention*. Jacksonville, FL: Paper presented at the Society for Information Technology & Teacher Education International Conference.

Baytak, A., & Land, S. M. (2010). A case study of educational game design by kids and for kids. *Procedia – Social and Behavioral Sciences, 2*(2), 5242–5246. http://dx.doi.org/10.1016/j.sbspro.2010.03.853.

Bonnardel, N. (2000). Towards understanding and supporting creativity in design: analogies in a constrained cognitive environment. *Knowledge-Based Systems, 13*(7–8), 505–513.

Bonnardel, N., & Zenasni, F. (2010). The impact of technology on creativity in design: an enhancement? *Creativity and Innovation Management, 19*(2), 180–191. http://dx.doi.org/10.1111/j.1467-8691.2010.00560.x.

Choi, W., & Repman, J. (1993). Effects of Pascal and FORTRAN programming on the problem-solving abilities of college students. *Journal of Research on Computing in Education, 25*(3), 290–302.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control, 2*(2), 137–167.

Cohen, J. (1988). *Statistical power analysis for behavioral science* (2nd ed.). Hillsdale, New Jersey: Lawrance Erlbaum Associates, Inc., Publishers.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: can they be used to measure understanding of computer science concepts? *Computers & Education, 58*(1), 240–249.

Field, A. (2009). *Discovering statistics using IBM SPSS statistics*. London: SAGE Publications Ltd.

Funke, J., & Frensch, P. (1995). Complex problem solving research in North America and Europe: an integrative review. *Foreign Psychology, 5*, 42–47.

Games, I. A., & Kane, L. P. (2011). Exploring adolescent's STEM learning through scaffolded game design. In *Proceedings of the 6th international conference on foundations of digital games* (pp. 1–8). New York: ACM. http://dx.doi.org/10.1145/2159365.2159366.

Gick, M. L., & Holyoak, K. J. (1980). Analogical problem solving. *Cognitive Psychology, 12*(3), 306–355.

Greeno, J. G., Collins, A. M., & Resnick, L. B. (1996). Cognition and learning. In D. C. Berliner, & R. C. Calfee (Eds.), *Handbook of educational psychology* (pp. 15–46). New York, NY: Simon & Schuster Macmillan.

Guzdial, M. (2004). Programming environments for novices. In S. Fincher, & M. Petre (Eds.), *Computer science education research* (pp. 1–16). The Netherlands, Lisse: Taylor & Francis.

Hambleton, R. K. (1990). Item response theory: introduction and bibliography. *Psicothema, 2*(1), 97–107.

Huber, O. (1995). Complex problem solving as multi stage decision making. In P. A. Frensch, & J. Funke (Eds.), *Complex problem solving: The European perspective* (pp. 151–173). Hillsdale, NJ: Lawrence Erlbaum Associates.

Hwang, G.-J., Hung, C.-M., & Chen, N.-S. (2013). Improving learning achievements, motivations and problem-solving skills through a peer assessment-based game development approach. *Educational Technology Research and Development*. http://dx.doi.org/10.1007/s11423-013-9320-7.

Isaacson, W. (2011). *Steve jobs*. New York: Simon & Schuster.

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development, 48*(4), 63–85.

Jonassen, D. H. (2004). *Learning to solve problems: An instructional design guide*. San Francisco, CA: Pfeiffer.

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: a case study on mathematics learning during design and computing. *Computers & Education, 73*(1), 26–39. http://dx.doi.org/10.1016/j.compedu.2013.12.010.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*(2), 75–86.

Klopfer, E., Roque, R., Huang, W., Wendel, D., & Scheintaub, H. (2009). The simulation cycle: combining games, simulations, engineering and science using StarLogo TNG. *E-Learning, 6*(1), 71. http://dx.doi.org/10.2304/elea.2009.6.1.71.

Li, Q. (2010). Digital game building: learning in a participatory culture. *Educational Research, 52*(4), 427–443. http://dx.doi.org/10.1080/00131881.2010.524752.

Li, Q. (2013). Digital game building as assessment: a study of secondary students' experience. *Developments in Business Simulation and Experiential Learning, 40*, 74–78.

Mayer, R. E. (1977). *Thinking and problem solving: An introduction to human cognition and learning*. Glenview, Illinois: Scott, Foresman and Company.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? The case for guided methods of instruction. *The American Psychologist, 59*(1), 14–19. http://dx.doi.org/10.1037/0003-066X.59.1.14.

Mayer, R. E., & Wittrock, M. C. (1996). Problem-solving transfer. In D. C. Berliner, & R. C. Calfee (Eds.), *Handbook of educational psychology* (pp. 47–62). New York, NY: Macmillan Library Reference.

Mayer, R. E., & Wittrock, M. C. (2006). Problem solving. In P. A. Alexander, & P. H. Winne (Eds.), *Handbook of educational psychology* (pp. 287–303). Mahwah, NJ: Lawrence Erlbaum Associates.

National Research Council. (1999). *Being fluent with information technology*. Washington, DC: National Academies Press.

OECD. (2003). *PISA 2003 Assessment framework: Mathematics, reading, science and problem solving knowledge and skills*. Paris.

OECD. (2013). *Test questions – PISA 2003*.

Papastergiou, M. (2009). Digital game-based learning in high school computer science education: impact on educational effectiveness and student motivation. *Computers & Education, 52*(1), 1–12.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books, Inc.

Papert, S., & Harel, I. (1991). Situating constructionism. In S. Papert, & I. Harel (Eds.), *Constructionism* (pp. 1–11). Norwood, NJ: Ablex Publishing Corporation.

Perkins, D. N. (1986). *Knowledge as design*. New Jersey, USA: Lawrance Erlbaum Associates, Inc., Publishers.

Polya, G. (1957). *How to solve it*. Garden City, NY: Doubleday/Anchor.
Reed, W., & Palumbo, D. (1988). The effect of the BASIC programming language on problem-solving skills and computer anxiety. *Computers in the Schools, 4*(3–4), 91–104.
Richards, K., & Wu, M. L. (2012). Learning with educational games for the intrepid 21st century learners. In P. Resta (Ed.), *Proceedings of society for information technology & teacher education international conference* (pp. 55–74). Chesapeake, VA: AACE.
Smith, K., & Boling, E. (2009). What do we make of design? Design as a concept in educational technology. *Educational Technology, 49*(4), 3–17.
Unuakhalu, M. F. (2009). Enhancing problem-solving capabilities using object-oriented programming language. *Journal of Educational Technology Systems, 37*(2), 121–137.
Wu, M. L., & Richards, K. (2011). Facilitating computational thinking through game design. In M. Chang, W. Y. Hwang, M. P. Chen, & W. Müller (Eds.), *Edutainment technologies, educational games and virtual reality/augmented reality applications* (pp. 220–227). Heidelberg: Springer Berlin.