Article

Supporting Children's Learning of Probability Through Video Game Programming

Journal of Educational Computing Research
0(0) 1–32

DOI: 10.1177/0735633115598492
jec.sagepub.com
SAGE

Yavuz Akpinar[1] and Ümit Aslan[1]

Abstract

Teaching programming and creating games have attracted much attention over the years, mostly the attention of curriculum developers and teachers. This study designed and developed a video game-based intervention, then investigated the effects of this intervention on middle school students' learning of probability concepts. In the study, the students learned and used Scratch as a game programming tool. Initially, they received hands-on learning activities on how to use Scratch, and then developed video games based on scenarios authored by the researchers. The study collected quantitative as well as qualitative data using two different measurement tools: probability achievement test and student project assessment rubric, respectively. The data revealed that students were able to learn and use Scratch and develop probability-related and probability-based algorithms that generate random results successfully. The effect of Scratch intervention on students' learning of probability was statistically significant. Findings were discussed in relation to similar studies reported in the literature. Finally, the study raised a set of further research questions in the Conclusion section.

Keywords

Game-based learning, secondary education, programming, probability

[1]Boğaziçi University, Etiler, Istanbul, Turkey

Corresponding Author:
Yavuz Akpinar, Boğaziçi University, ETA-B Blok, 508, Kuzey Kampüs, Etiler, Istanbul, Turkey.
Email: akpinar@boun.edu.tr

Introduction

With the advancements in technology and wide availability of computing tools, using programming and game creation as a context to teach curricular domains such as mathematics, English, and science as well as higher order thinking skills has been the subject of interest of researchers (e.g., Akcaoglu & Koehler, 2014; Baytak & Land, 2011; deHaan, 2011; Hwang, Hung, & Chen, 2014; Ke, 2014). For example, LOGO, the first visual programming language designed for educational use, enabled users to control a mechanical robot to create a dialog between the computer and the turtle (Papert, 1980; Papert & Solomon, 1971). Using tools like LOGO, students enter a microworld in which they can develop various models and test them repeatedly; so that they can see what is working and what is not working (Edwards, 1991; Papert, 1980; Papert & Harel, 1991).

In some research studies, computer game design and programming activities were used in order to provide such microworlds and foster a culture of self-motivation, self-expression, creativity, and knowledge construction. The study conducted by Harel and Papert (1990) examined students' learning of fractions with LOGO versus with regular classroom activities. The study revealed that studying LOGO and fractions together was more successful than studying each in isolation. Likewise, Edwards's (1991) study with sixth to eighth grade middle school students using a geometry microworld, which was a set of simple LOGO commands, effectively helped students correct their tendency to overgeneralize symbolic patterns in geometric transformations. Also, Resnick's (1996) study with StarLogo programming language, as an environment for exploratory learning in statistics and randomness, provided a tool for developing an understanding of decentralized parallel situations about which people have misconceptions and difficulty in understanding. Other researchers have also tried to teach mathematics subjects such as ratio and proportion, and geometry in more intuitive and motivating LOGO-based learning environments where children can challenge their mathematics knowledge, learn new things, and use them to create personally meaningful artifacts (Clements & Battista, 1989; Clements & Sarama, 1997; Hoyles & Noss, 1992; Resnick, 1997). However, studies on learning by programming with LOGO could not provide sufficient amount of evidence to break into curricular units and become a permanent actor in the field of education.

Learning by Designing Games and Multimedia

With advancements in media technologies and the advent of the internet, many different tools of learning by programming (similar to LOGO) and learning by designing were developed and studied. For example, Mor and Sendova (2003) studied the effects of using Toontalk, an animated programming language, on primary school students' attitudes toward mathematics and mathematical

achievement and showed that students were able to develop insights into the relationship between modeling and underlying mathematical structures. Furthermore, many researchers focused on giving students the role of an active designer (e.g., Calder, 2010; Harel & Papert, 1990; Kafai, Ching, & Marshall, 1997; Kafai, Franke, Ching, & Shih, 1998; McCue, 2011; Robertson & Howells, 2008). Kafai et al. (1998) argue that game design activities provide students with a learning environment in which they can build on and challenge their existing understandings, engage in relevant and meaningful learning contexts, and develop connections between the curriculum subjects and the real world contexts. In such activities, students get an opportunity to think like planners, problem solvers, and designers in a continuum. Accordingly, in order to examine the effects of multimedia design on learning, Kafai et al. (1997) conducted a study with fifth and sixth graders who created interactive multimedia resources about astronomy using microworlds LOGO programming environment. They found that students' understanding of astronomy concepts and LOGO programming were developed significantly. In another study by Kafai et al. (1998), the effects of designing video games with pen and paper on understanding of fractions were analyzed. At the beginning, content of fractions and game ideas were isolated from each other in the projects of most participants, but after various design activities, participants were better able to integrate fractions content into their ideas. Later, Robertson and Howells (2008) conducted an exploratory field study with sixth graders using a game engine software, which enabled them to create characters, change landscapes, and write interactive dialogs in a virtual 3D environment. They found that during the intervention, children displayed motivation and enthusiasm for learning, determination to reach a high standard of achievement, and independent learning skills. In addition, students were able to relate and apply their learning to new situations. In a more recent study, deHaan (2011) investigated game construction activities in students' learning of English using the RPG Maker software. It was reported that the projects motivated and challenged the students, provided opportunities for authentic discussions in the target language, and offered students concrete language, technology, teamwork, and creative experiences. In short, studies on game and multimedia application design provided promising results.

Creative Computing and Scratch

While evidence for teaching curricular units with programming tools is scarce, other tools and approaches for learning by programming and design were introduced. Scratch, for example, is another programming tool developed by Lifelong Kindergarten Group in MIT Media Lab (Resnick, 2007; 2012; Resnick et al., 2009). It visualizes the computational process but with a different approach. Built upon the principles of LOGO, Scratch provides programming blocks

similar to puzzle pieces to assemble in order to create algorithms that read like spoken language without worrying about syntax errors. Arguing that most learning environments today are not designed to help students develop creative thinking skills, Resnick (1998, 2007) claims that the traditional kindergarten approach in which children are constantly designing, creating, experimenting, and exploring is well matched to the needs of the current society, and it should be extended to learners of all ages. Resnick proposes a learning environment in which children continuously imagine, create, play, and share and reflect (see Figure 1).

The first step of the spiral cycle, *imagine* represents a learning environment, either digital or physical, in which provided learning tools and materials are powerful enough to be used in multiple ways leaving more room for children's imaginations. The second step, *create*, is about providing children with opportunities to design and create things by themselves. Third, *play* stands for integrating play, design, and learning in a way that children continuously experiment, explore, and test the boundaries. As the fourth step, *share* means creating a learning environment in which children can share their constructions with others so that they can become engaged with both construction process and the community. Lastly, *reflect* emphasizes the importance of critical reflection on the ideas that guided the design, or strategies to refine and improve the design or connections by using underlying scientific concepts and related real-world phenomena. However, reflection is not the last step of this process. In contrast, this

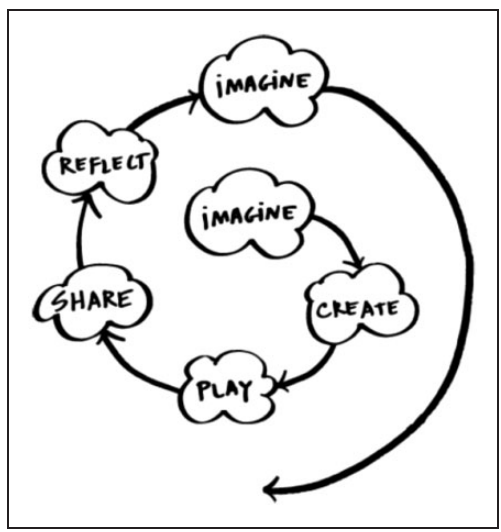

Figure 1. The creative learning spiral (Resnick, 2007).

process helps to generate new ideas, triggering another cycle that makes it an iterative process.

Additionally, Resnick (1998, 2006) argues that children use computers mostly in a passive and consumption-based way. He argues that in order to realize the real potential of computers, we should start thinking of them more like paintbrushes rather than televisions. In other words, we should consider computers as a new medium for creative design and expression rather than a machine to retrieve information. Instead of taking traditional classroom activities and simply reimplementing them on the computer, technology should be used to create learning environments in which children, as active participants, are engaged in creative tasks. In parallel with Resnick's ideas, Brennan (2011) introduces the concept of creative computing, a design-based approach to engage young people in the creation of computational artifacts, to support young people's development as computational thinkers—individuals who can use computational skills in their everyday lives.

Place of Creative Computing in Learning Mathematics

Even though Scratch has been actively used in the scene of education as a creative computing tool, most applications in the literature focus on its effects on computational thinking, science learning, digital literacy, and perception of computer-based majors (e.g., Baytak & Land, 2011; Blau, Zuckerman, & Monroy-Hernandez, 2009; Brennan, 2011; Kafai, Peppler, & Chiu, 2007; Maloney, Peppler, Kafai, Resnick, & Rusk, 2008; Meerbaum-Salant, Armoni, & Ben-Ari, 2010; Peppler & Kafai, 2007; Wolz, Stone, Pulimood, & Pearson, 2010). Unfortunately, there is limited number of studies focusing on applications of Scratch in mathematics education. The results of the studies that have been conducted show controversial results about the effects of creative computing on learning mathematics. For example, Boyer (2010) investigated the effects of Scratch programming on learning when fifth graders design multimedia artifacts that demonstrate an understanding of geometric solids. In contrast to studies conducted with LOGO or its derivatives (e.g., Edwards, 1991; Harel & Papert, 1990; Hoyles & Noss, 1992), Boyer's study had mixed results showing that learning is not guaranteed when students are engaged in design tasks on Scratch. Further, Taylor, Harlow, and Farret (2010) investigated the effects of using Scratch on mathematical thinking. They found that Scratch programming activities improved student motivation and gave them an opportunity to explore and use sophisticated mathematical concepts. Recently, Lewis and Sarah (2012) found a high positive correlation between students' scores on a standardized test for mathematics and their scores on Scratch programming quizzes after a Scratch summer program for sixth graders. Finally, Ke (2014) investigated the potential of game making activities with Scratch in facilitating design-based math learning for middle school students. According to Ke, the students

developed significantly more positive attitudes toward mathematics, and the game design processes helped to activate children's reflection on everyday mathematical experiences, and mathematical thinking and content experience were intertwined within the process of computer game programming. However, the study did not provide evidence on how much math content the students learned. While there are inconclusive findings about the relationship between programming and mathematics achievement, there is an understanding that learning programming has a potential to be a new tool for reflection and problem solving in a learning domain (e.g., probability).

The situation is not different in analyzing students' progress in creative computing activities. Previous studies used a number of different tools to analyze students' projects created by using Scratch. Boyer (2010) created a spreadsheet and recorded the number of sprites, backgrounds, scripts, and command blocks used by students studying the properties of geometric solids in their projects. Also, he recorded his reflections on students' progress each week. In another investigation of the effects of Scratch programming, Baytak and Land (2011) used an evaluation rubric with four dimensions: game genre, graphics and character development, control options, and duration of the game to analyze projects. Besides, they analyzed students' use of different Scratch programming concepts by utilizing the methodology developed by Maloney et al. (2008). Later, Denner, Werner, and Ortiz, (2011) created a coding scheme, based on International Society for Technology Education, National Education Technology Standards for students, and Martin, Walter, and Barron (2009) created by another similar scheme (cited in Denner et al., 2011). They coded each game in three categories (programming, documenting and understanding software, and designing for usability) and 24 subcategories. Unfortunately, the number of studies is low, and none of these studies were conducted on learning and teaching probability. Still, they were helpful in choosing a proper analysis method in this study.

Creative Computing Activities, Probabilistic Thinking, and Reflective Problem Solving

Piaget and Inhelder (1975) initiated some research efforts on intuitive sources of probability in the 1950s. Through a series of studies, they concluded that children start developing the idea of "chance" approximately at the age of 6. Furthermore, they concluded that children between the ages 6 and 14 have no systematic approach toward generating a list of probabilities and, therefore, they do not have any mathematical maturity to make an abstract model of a probability experiment. Their findings sparked a still on-going debate on how children develop the idea of chance and probabilistic concepts, as well as the possibility of bypassing Piagetian stages (e.g., Biehler & Pratt, 2012; Davies, 1965; Goldberg, 1966). Although this debate is still inconclusive, it is safe to

infer from the literature that the age period between approximately 6 and 13 is critical in children's learning of probability concepts and development of probabilistic thinking. Another important issue in probabilistic thinking research is the potential negative effects of schooling. For example, in his studies with primary school children, Fischbein (1975) showed that older children hesitated more often in uncertainty tasks and failed more often than younger children (p. 163). He argued that traditional schooling prioritizes deterministic aspects of life and mathematics and neglects nondeterministic ones. Similarly, Wilensky (1997) argued that developing deterministic-only mindsets through traditional schooling hurts children's development of probabilistic intuitions and results in a kind of epistemological anxiety. Even though research on this particular issue is scarce and still debated even four decades after Fischbein's initial studies (Biehler & Pratt, 2012), it can be assumed that children may be responsive to appropriate instruction on probability concepts. For example, Drier (2000) recommends a microworld as a virtual manipulative to provide an alternative to the traditional hands on experiences in the instructional process.

It is generally accepted that mathematics education, including probability, is not simply about mathematical concepts and skills but also its processes (Hoyles & Sutherland, 1992). These concepts and skills are adapted and employed in the process of a variety of situations. Although different problem-solving strategies have been taught in mathematics classrooms (e.g., Bransford & Stein, 1984; Polya, 1954; Schoenfeld, 1985), more attention has been paid to encourage students to shift from a product-oriented approach, which is concerned only with a superficial involvement with the problem, to one that is more reflective and demanding effort and time commitment. Hoyles and Sutherland (1992) argued that such a shift in the didactical relations in mathematics classrooms might increase student autonomy and responsibility. This shift implies more group interaction and interaction with thought-provoking tools and manipulatives, including virtual toolkits, and argumentative collaboration among students and between student and toolkits as a vehicle for developing problem-solving skills. Researchers also stress the need for a structural knowledge base for problems and contextual influences on problem-solving approaches (e.g., Brown & Walter, 1983; Hoyles & Sutherland, 1992; Lave, Murtaugh, & de la Rocha, 1984; Rosenbaum, 2009). For instance, Hoyles and Sutherland (1992) highlighted that when a student knows about the context in which problem is embedded, it will be easier to solve compared with decontextualized problems.

Processes of mathematical problem solving were comprehensively codified by Polya (1954) in his classical work proposing a generic four-step problem-solving strategies: (a) understanding the problem, (b) devising a plan, (c) carrying out the plan, and (d) looking back at work. Much later, Bransford and Stein (1984) developed a problem-solving model with five steps briefly known as IDEAL: (a) identifying problems and opportunities, (b) defining goals, (c) exploring

possible strategies, (d) anticipating outcomes, and (e) looking back and learning. In a similar vein, but in a narrower context, as outlined earlier, Resnick (2007) proposed a spiral learning cycle, which requires an iterative process to be designed to reinforce critical thinking in the context of problem solving. Resnick's spiral model is an extension of Papert's microworld model of learning through LOGO programming (Papert, 1980), where learning is defined as an active process of planning, developing, reflection, and debugging. In Papert's model, the planning phase is parallel to Resnick's imagine phase, the developing phase to create and play phases, the reflection and the debugging phases to share and reflect phases. When transferring problem-solving skills developed in computer programming to mathematics is considered the two should not be separated but be considered together in task, context, and activity aspects. Further, the programming activity and the outcomes of the activity should allow students to externalize their own thinking and reflect on it. This sort of metacognition in the problem-solving process can be developed when content, context, and task are appropriated for a student's level of progress and supported through computer artifacts such as the Scratch environment. Besides, the learning environment to be designed around Scratch may supply facilities to students to select strategies, to try out ideas, to initiate challenging solutions, to organize, to sequence, to implement, and to evaluate reflectively.

Research Problems

In this study, the intention is to bring foundational principles of Scratch into mathematics education in order to build an engaging learning environment for students to study an abstract and unpopular mathematics subject, probability. Many software tools for learning probability are either too complex to learn or too simple to be helpful; so students have very limited opportunities to understand the underlying concepts (Abrahamson, Janusz, & Wilensky, 2006; Bar-On & Or-Bach, 1988; Konold, 1995; Memnun, 2008; Resnick & Wilensky, 1998; Shaughnessy, 1992). Scratch, while it does not bring any ready-made tools for learning probability, provides a development environment which is easy to learn, and intuitive for young students. Once students learn basic tools of Scratch, they can develop their own applications on probability such as simulations, games, and experiments. By trying to develop algorithms for their projects, manipulating these algorithms and changing structures, they can improve their probabilistic thinking. Moreover, such a learning environment enables educators to include explicit and formal aspects of probability more easily than traditional teaching practices, and this may increase students' reflective thinking in activities. This study aims to explore the effects of middle school students' development of video games with Scratch on their achievement of independent events in probability.

Methodology

Design and Sample

The design of the study is a one-group pretest-intervention-posttest quasi-experimental design. There is not any random assignment of groups of individual members in this design (Campbell & Stanley, 1963). This methodological design was used to compare the extent of change in students' knowledge of probability. The research was conducted in a school in a metropolitan city of Turkey. Participants of the study were 18 fifth grade and 12 sixth graders (aged 12–14). This school accepts students without any entrance exam and does not classify students according to their ability. In the study, students were not classified according to their grade levels, and they participated in meetings together. In terms of prior probability knowledge, based on students' scores in a probability achievement test (PAT) administered before the experiment (fifth graders' mean scores was 3.56 [$SD = 2.33$] and sixth graders' mean scores was 3.25 [$SD = 2.77$]), the group was heterogeneous.

Instrumentation

To answer the research question, quantitative data were collected through a PAT, and qualitative data were collected through a student project assessment rubric (SPAR).

Probability achievement test. To measure participants' achievement in probability, a PAT developed by Kavasoglu (2010) was used. Questions of the instrument were developed to measure objectives in probability curriculum of the middle school. The probability test (with a 0.90 reliability coefficient) included 32 multiple-choice questions with four alternative answers. For this study, the test was split into two parallel tests as a pretest and a posttest according to the objectives of the unit. Three items of the pretest are given in Figure 2.

Student project assessment rubric. In addition to quantitative analysis of students' achievement in probability, their Scratch projects were collected to analyze their experiences qualitatively for two reasons. First, there is no quantitative measurement tool to ensure that students learned Scratch as a tool. If students did not learn Scratch well enough, it may have become a demotivating factor in learning probability, as opposed to the intended fostering and encouraging effects. Second, it is important to check if students were able to transfer their probabilistic thinking skills to Scratch projects. To come up with a framework for qualitative analysis, similar studies conducted in recent years were reviewed. In the light of the review, summarized earlier, an assessment rubric was designed to analyze students' projects. After the design phase, the first draft of the rubric was revised according to the feedback given by four mathematics

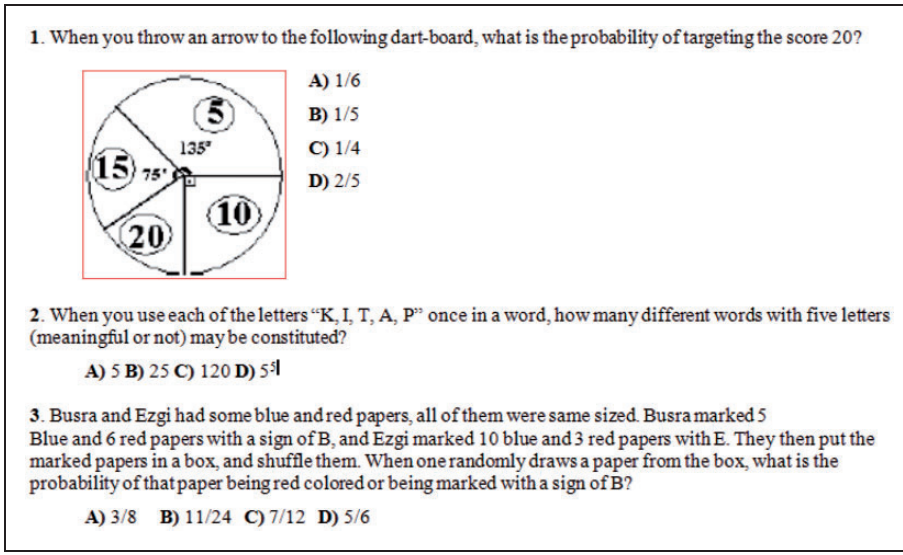

Figure 2. Three sample questions of the probability achievement test.

educators and three educational technology experts. The final form of the rubric (Table 1) contains two dimensions: developing video games and developing strategies for solutions of probability problems (for details see Aslan, 2014). All items of the rubric are rated as not evident, insufficient, successful, and creative.

Procedure

The pretest data for PAT was collected 1 week before the first activity. Then, students learned how to develop projects in Scratch for 4 weeks (for details, see Tables 2 and 3 for activities of the intervention). As part of a mathematics applications course, we conducted nine workshops, once in a week. Each workshop consisted of two 40-min sessions with a 10-min break between the two. The first four workshops of the study included 30 min of hands-on Scratch programming instruction and 50 min of developing games, followed by discussions with friends and teachers. The students took part in game development and programming activities on an individual basis. In the following 5 weeks, the researcher gave students five different tasks each week. All tasks were divided into two phases. At the beginning of the activities, the students developed projects based on the task and their imaginations. After about 40 min of development, their math teacher conducted discussions with students for 10 min. Following the discussions, the students were asked to make several modifications or additions in their projects. The role of the researcher was to help students learn

Table 1. Student Project Assessment Rubric.

		Not evident	Insufficient	Successful	Creative	Researcher comments
Developing video games	Usage of sprites					
	Usage of Scratch blocks					
	Development of algorithms					
	User input					
	User feedback					
	Variables					
	Overall project completion					
Developing strategies for solutions of probability	Representation of sample space					
	Experiment design					
	Experiment implementation					
	Multiple experiment simulations					
	Outcome representation					

Scratch programming language and assist them in accomplishing tasks. While students were working on tasks, he continuously monitored their progress and helped them whenever they needed. He was mostly passive in probability discussions. He provided guidance about Scratch to help students when the mathematics teacher asked them to make adjustments in their projects. The last five projects developed by the students were collected for data analysis through SPAR. Three weeks after the last workshop, the students were given the posttest (the tasks were tried out in a pilot study with different students, for details see Aslan, 2014).

Data Analysis and Results

To explore the possible effects of interventions on students' PAT scores, first, effect size measures were conducted. Then the PAT score data were checked for normal distribution with Kolmogorov–Smirnov and Shapiro–Wilk tests. Because the PAT scores did not show a normal distribution, nonparametric tests were conducted. Students' scores in pre- and post-PATs are summarized in Table 4. The minimum score is 0 and the maximum score is 14 in PAT. In the

Table 2. The Study Activities 1 to 4.

1	Introduction to Scratch	Characters, backgrounds, movement blocks, control blocks	Aquarium animation	Students will learn how to use Scratch. They will develop an aquarium animation in which at least three fish flocks around the screen programmatically.	After developing the aquarium, the teacher will conduct a discussion about the movements of fish and the mathematical concepts they need to know in order to change the movement of fish. Then she or he will draw some shapes to the board and ask students to make their fish move in such shapes. In the end, students will again discuss the mathematics concepts they utilized and calculate the distance their fish traveled.
2	Introduction to Scratch	Costumes, look blocks, events, user input	Loves me, loves me not	Students will learn how to change costumes and appearance of characters. They will also learn how to use keyboard input in their projects. They will develop a "loves me, loves me not" game in which the player will press a keyboard button to pick up the leaves of a flower one by one.	After finishing their initial projects, the students will discuss the relevance of fractions with their flowers and leaves. Then the teacher will write some fractions on the board and ask students to change their projects according to these fractions.

(continued)

Table 2. Continued

3	Advanced Scratch	Variables, loops, logic	Lottery simulation	Students will learn how to use variables, loops, and if-else blocks. They will develop a lottery simulation which can be manipulated by changing values of variables such as digit count or max number.	The teacher will discuss sample spaces with students and ask them about some certain events and some impossible events. Students will be asked to make experiments with some given variable values. Lastly, they will be asked to manipulate their projects in order to make 1,000 experiments at once and find out the probability of each number.
4	Advanced Scratch	Sensing, variables, control blocks	Pet feeding	Students will develop a pet feeding game, in which the user controls an animal with keyboard, tries to eat delicious foods, and avoid bad ones. The animal will grow or shrink according to the score of the player.	The teacher will conduct a discussion about ratio concept and its relevance to this task. Then she or he will write some ratios on the board and ask students to make their animals grow or shrink according to the given ratios.

Table 3. The Study Activities: 5 to 9.

5	Game development	Basic events	Heads and tails	Students will be asked to develop a heads and tails game. There will be a section to make the choice and then a button to flip the coin randomly.	The teacher will use one of the students' projects and play the game with the whole class. Then she will ask them to flip the coin 20 times, note the results and discuss the probability of the event. Furthermore, she will collect the results of all students and calculate the total number of heads and tails on the board. In the end, students will be asked to convert their projects into a simulation tool in which one can flip a coin 10, 100, and 1000 times.
6	Game development	Basic and independent events	Dice game	Students will develop a dice game in which two players roll a dice in turns. The first player reaching 100 points will win the game.	The teacher will conduct a discussion on probability of getting a number after rolling the dice. She will ask if an event affects the result of its successor. In the end, students will calculate the probability of getting each number after rolling a fair dice.
7	Problem solving	Basic principles of counting	Finding routes	Students will develop a project to estimate the number of different routes of a car on given path, then answer a relevant probability question.	The teacher will conduct a discussion on basic principles of counting with students and ask them to add another city to their projects. Then she will calculate the new routes by discussing with students.

(continued)

Table 3. Continued

8	Game development	Basic events	100 m running	Students will develop an Olympic 100 m running game. There will be six runners and a dice in the game. When the player presses a button, the dice will roll and the runners will run accordingly. For example, if upper face of the dice shows 1, the first runner will go 10 steps and if upper face of the dice shows 6, the sixth runner will go 10 steps.	The students will be asked to form groups of six and play the games together. Then they will be asked to convert their games to a simulation in which one can run the game 10, 100, or 1000 times. The results will be stored in variables.
9	Problem solving	Basic events	Wheel of fortune	Students will develop a project to simulate a wheel of fortune, then answer a relevant probability question.	After developing the game, students will be asked to convert their projects into a simulation in which the wheel can be turned 10, 100, or 1000 times. The results will be stored in variables. Then the teacher will conduct a discussion in the classroom about the results.

Table 4. Data for Students' Probability Achievement.

	Pretest		Posttest		Effect size	
n	Mean	*SD*	Mean	*SD*	Cohen's *d*	
30	3.43	2.47	5.57	2.11	0.93	Large

pretest, the students' scores ranged from 0 to 10. On the other hand, all students were able to answer at least two questions in the posttest, in which the maximum score was again 10.

Probability Achievement

To test whether the intervention had any effect on students' probability achievement, first Cohen's *d* values for the PAT scores were calculated. To avoid possible misleading results of significance testing in examination of effects using small sample sizes, effect size estimation is suggested (Cohen, 1990). The result showed a large effect ($0.80 < d < 1.30$) of intervention on students' probability achievement (Table 4). That was confirmed by a Wilcoxon signed rank test ($Z = -3.937$; $p < .0001$), as it concluded that there is a statistically significant difference between students' scores of probability achievement pretest and posttest showing a positive effect of the intervention on students' probability achievement.

Analysis of Projects Developed by Students

As stated earlier, projects developed by students in the last 5 weeks of the intervention were collected at the end of the workshops. All projects ($n = 115$) were rated by two independent raters, knowledgeable both in probability and Scratch, using the SPAR. Mean score for each item of the projects was calculated, and then accordingly, properties of the projects were interpreted.

Developing video games. Analysis of the students' projects (Table 5) revealed that the students comfortably maneuvered Scratch sprites in their projects. Only one of the collected projects did not include any sprites, and just two of them were marked as insufficient. On the other hand, only three of the collected projects included a creative usage of sprites. In short, data shows that the majority of the students stuck to the task and did not need to alter it very much.

The students capably identified and used necessary command blocks for their projects. None of the projects included creative usage of command blocks, which may be due to the similarity of tasks in terms of generating random results. It is important to keep in mind that although the item for development of algorithms does not judge development of random experiments in Scratch, seeing that

Table 5. Scratch Components Used by the Students.

	Not evident		Insufficient		Successful		Creative	
in Students' Projects;	*f*	%	*f*	%	*f*	%	*f*	%
Usage of sprites	1	0.87	2	1.74	109	94.78	3	2.61
Usage of Scratch blocks	2	1.74	2	1.74	111	96.52	0	0
Development of algorithms	2	1.74	6	5.22	103	89.57	4	3.48
User input	2	1.74	2	1.74	83	72.17	28	24.35
User feedback	3	2.61	51	44.35	58	50.43	3	2.61
Variables	5	4.35	45	39.13	60	52.17	5	4.35
Overall project completion	2	1.74	5	4.35	104	90.43	4	3.48

students had little difficulty in developing algorithms for their projects is a good indicator that students learned Scratch programming well enough. Students' utilization of user input methods (keyboard, mouse, webcam, etc.) in their projects were mostly assessed as successful, as well. Moreover, almost 25% of projects included creative user input options. We concluded that students knew how to get user input in Scratch well enough to develop basic probability applications. While students were able to utilize user input blocks in their projects, analysis of their projects indicates that there are problems in their preference of user feedback. Considering almost half of the projects were found to be insufficient in terms of user feedback, one can infer that students were unable to develop user feedback in Scratch or they were unaware of the importance of user feedback. It is safe to say that being able to develop algorithms for probability experiments is more important than reflecting the results on the screen. However, it is essential for students to implement on screen feedback in their projects because of the nature of the process of application development, which requires running the code, checking the results, making arrangements, and starting a new debugging cycle by running the code again. Therefore, being able to see the results of their own probability experiments is very crucial for the students.

Despite the lack of proper user feedback in 54 student projects and insufficiency of almost 40% of students' projects in terms of usage of variables, the latter is not actually dramatic because some activities such as seventh and eighth week activities did not require any usage of variables at all. Even so, it is still considered as a negative result. On the other hand, overall project completion is the most important item of developing video games dimension of SPAR because it judges internal consistency of a students' project. In other words, it checks if a student was able to identify necessary visuals, command blocks, algorithms, user input methods, user feedback methods, and variables to complete her project successfully. Fortunately, only a small number of (7/115) projects were found to

Table 6. Probability Components in the Students' Projects.

	Not evident		Insufficient		Successful		Creative	
In students' projects	*f*	%	*f*	%	*f*	%	*f*	%
Representation of sample space	1	0.87	4	3.48	110	95.65	0	0
Experiment design	2	1.74	2	1.74	99	86.09	12	10.43
Experiment implementation	3	2.61	4	3.48	106	92.17	2	1.74
Multiple experiment simulations	3	2.61	37	32.17	74	64.35	1	0.87
Outcome representation	2	1.74	7	6.09	101	87.83	5	4.35

be not evident or insufficient. Hence, it is concluded that students were able to start and finish a Scratch project.

The analysis of students' projects according to SPAR demonstrated that they had enough design and programming skills required to develop video games in Scratch based on basic events in probability. However, it has to be noted that programming skills are advanced through developing different projects under different conditions, and a complete program is produced with the support of others. The students completed programming activities only in an abstract domain of probability. When they conduct programming to provide computational solutions to less abstract domains or problems, many of programming concepts may be more easily mastered by them.

Developing strategies for solutions of probability problems. To begin with, students' representation of sample space in their projects was analyzed (Table 6). Understanding of sample space is very important in a project because if a student fails in doing so, she or he would fail in four other dimensions, too. Unsurprisingly, there is not a project that is marked as creative because changing the sample space of a probability experiment would generate unfair results. On the other hand, only 5 of 115 projects had problems in representing the sample space of the experiment. Therefore, it may be concluded that the students were able to determine sample space of probability experiments given in the tasks and convert them into visuals (sprites, costumes, texts) in Scratch. Another pleasing result is the students' experiment design in Scratch; 86% of the students' projects were marked as successful, and 10% of the projects were marked as creative. It shows that students were able to convert a probability task or an experiment to a video game project.

Experiment implementation in students' projects was found to be mostly successful, too. Probability experiments in 108 out of 115 projects generated random results as expected, showing that students were able to transfer their learning of basic events to algorithms. On the other hand, it is important to note that this item assesses the implementation of algorithms solely and does not take

representation of results into account. Considering the notable value of conducting multiple experiments independently, seeing results cumulatively and comparing the results of these experiments with others' results in learning probability, a partly negative result was found in students' utilization of multiple experiment simulations. Even though in four out of five tasks (except Task 7) students were asked to implement multiple experiments in their projects, at least as a follow-up activity, unfortunately 40 out of 115 projects in this regard were marked as unsuccessful by the evaluators.

Lastly, the majority of students' projects in terms of representation of outcomes were successful. Even though some students had difficulty in conducting multiple experiments or showing results of them, they were still able to show results of particular experiments visually. In conclusion, except running experiments more than once automatically, the students were able to develop strategies for solutions of probability problems in video games that they developed using Scratch.

Reflections on selected student projects. Based on evaluation of students' projects, all projects were categorized as insufficient, successful, or creative. To illustrate the process of evaluation of students' Scratch projects and exemplify the magnitude of their programming skills, three projects were selected as case studies and analyzed within the context of SPAR, and a pseudonym has been substituted for the actual name of the student.

Development of probability race game. Aylin is a fifth grade female student, who developed a probability racing game (see Figure 3) according to the task of the eighth week lesson. The task was to develop a racing game based on

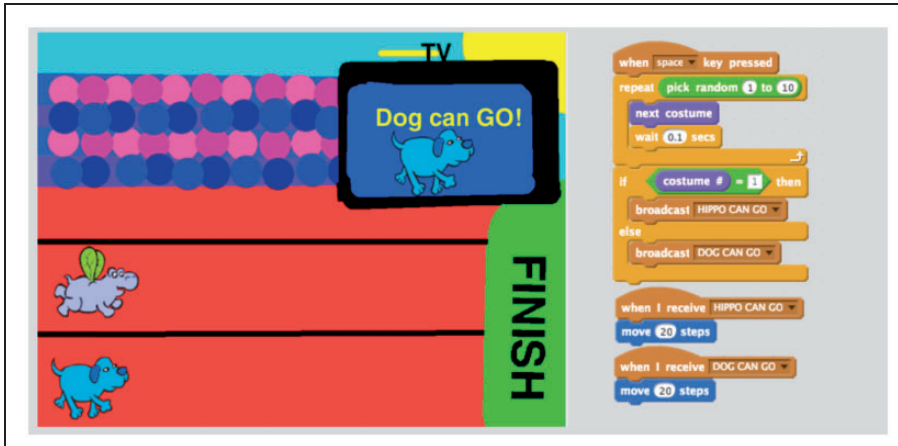

Figure 3. Screenshots from Aylin's project: the game and its Scratch blocks.

outcomes of a probability experiment. To narrow down the scope and make development easier for the students, they were asked to create a coin toss experiment with two races. If the result is heads, the first racer will go one step; otherwise, the second racer will advance. The one who reaches the finish line first will win.

In terms of developing video games dimension of SPAR, Aylin's project is enjoyable and creative. She used four different sprites with different costumes in each of them and created a background containing a stadium filled with fans, running lanes, and a television screen. Her project consists of 12 different Scratch blocks, all of which are relevant to the project. In short, her usage of blocks and algorithms are successful. She managed to develop an algorithm for running a probability experiment identical to heads-tails, and another algorithm to move the racers according to the result of the experiment. User input, which is pressing the spacebar to start a new experiment, was successful as well because the task did not expect students to do more. User feedback in her project is creative because there are multiple feedbacks: text feedback for result of the experiment on the TV screen, the image of the winner character on the screen, and the movement of the winner character. In this particular project, variables are marked as "not evident" because she did not create any of them. Yet, it does not affect completion of the overall project, as it is possible to complete this project without using variables. In the end, this project is evaluated as successful in relation to overall project completion, as it reflects the student's ability to use Scratch' visual tools, blocks, and algorithms effectively. As well as her competence in developing video games, Aylin's project reflects her competence in developing strategies to solve probability problems. Instead of using a coin or coins in the project, she designed a television with two different costumes. The first costume shows the winner as the dog and the second shows the winner as the hippo, (the racers). The system works identical to a coin toss experiment. Thus, both representations of the sample space and the experiment design in her project are creative. Rather than putting a coin on the screen isolated, she created an authentic scene for her project. Experiment implementation is successful, and her algorithms work as expected. Multiple experiment simulations are successful, too, because there is a need to run more than one experiment, and the user can see the results of the experiments cumulatively as the racers advance. Besides, the position of each character is reset at the beginning of the race. With a properly working TV to represent outcomes, texts, and advancement of characters all work together successfully, Aylin's project is a good example for a creative project in terms of outcome representation. To sum up, Aylin's project is a very good example of how creative a probability project can be developed by students using Scratch. Considering that her pretest score was 3 and posttest score was 8 in PAT, it is concluded that Aylin made a good progress in terms of probability knowledge.

Development of heads and tails experiment. The second case is a heads and tails experiment developed by a fifth grade male student, Murat, as the task of the fifth week activity. In this task, students were asked to develop a simple heads and tails experiment project. After the initial development, the researcher conducted a discussion with students about the results of a particular experiment, successive trials and independent events. The responses provided by students were written to the board, and cumulative ratio of heads to tails was calculated. Afterwards, the students were asked to develop a project that enables the user to run experiments many times such as 50, 100, 500, or even 1,000 times.

Regarding the first dimension of the SPAR, developing video games, Murat's usage of sprites is marked as successful because he used only one sprite, two costumes, and a background which is enough to complete the project correctly. Second, he properly used eight different command blocks, and two nested algorithms: one for tossing the coin randomly and another for tossing it 100 times, and recording the results in variables. Both algorithms run as expected, though there are not any creative additions. In terms of user input, Murat's project (Figure 4) can be assessed as successful even though there is just one input: pressing the space bar. When space bar on a keyboard is pressed, the game starts running heads and tails experiment for 100 times, which is adequate for the task. In terms of user feedback, on the other hand, his project is insufficient: there is only feedback in terms of variables. but the user is unable to view the outcomes of individual experiments separately. There are also no instructions for the user about how to use the application. The project is insufficient in terms

Figure 4. Screenshots from Murat's project: the game and Scratch blocks used in his game.

of variables, as well. There are two variables in the project, one head and one tail, as expected by the task, but the variables are not reset at the beginning of each experiment. Lastly, checking overall project completion, Murat's project is considered as successful because sprites, command blocks, algorithms, user input, and variables of the project are employed almost as expected.

In relation to developing strategies for solutions of probability problems, Murat's project is successful in representing sample space of the experiment. There is one coin character with two costumes in it: one head and one tail image. The coin cannot be head and tail at the same time. Experiment design of the project is successful, as well. Even though he did not put any instructions on the screen for the user, he draws a human hand flipping the coin instead of showing an isolated coin. His implementation of the experiment is successful, too, because his algorithms work as expected, and his project generates random results. Moreover, Murat successfully developed algorithms for running probability experiments more than once and showed the results as changes in variables decorously. Yet, representations of outcomes in Murat's project are insufficient because of the following reasons: First, the project does not reset variables properly. Second, one cannot see the results of individual experiments. Third, there is no on screen feedback for the results of individual or multiple experiments. To sum up, Murat's project is a good example of a successful project, which follows the worksheet without much deviation. Even though some parts of the project need improvements, he transferred his understanding of this particular probability experiment into a video game. His quantitative probability achievement score is 2 in the pretest and 6 in the posttest. Putting all of these together, it is fair to say that Murat improved his probability knowledge, though there was still room for improvement.

Development of wheel of fortune experiment. Emek, a fifth grade male student, is the developer of the last case project. His project (Figure 5) is a probability simulation developed to understand a formal mathematics problem in the ninth week. Emek developed a gift lottery project in which the user receives a white box or a red box as a Christmas gift. First, in terms of usage of sprites, he correctly used an arrow, a wheel, and a background text for his project. Second, he successfully integrated 10 different command blocks into his project. Third, he developed two different algorithms: an algorithm to turn the wheel randomly and another algorithm to determine the result of the experiment. Both algorithms work properly. User input in his project is considered successful, too. However, user feedback is insufficient in Emek's project because there is no textual or visual feedback to be provided to the user. He uses variables to record the results of experiments but does not inform the user about the results of particular experiments. Use of variables in his project is successful because he keeps the results of experiments cumulatively and resets both variables at the beginning. Emek's overall project completion is considered as successful because

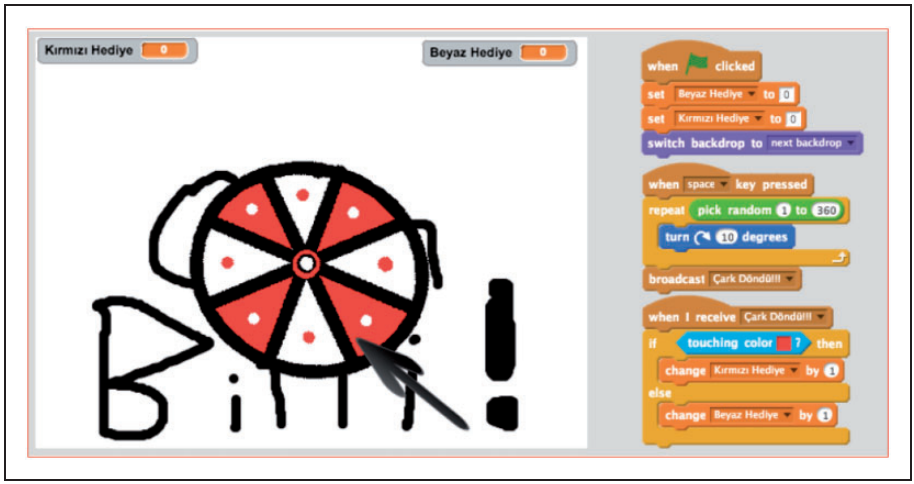

Figure 5. Screenshots from Emek's project: the game and Scratch blocks used in his game.

the project contains all necessary sprites, algorithms, input methods, and variables. In terms of developing strategies for solutions of probability problems, Emek's project is unfortunately not a good example: The wheel was divided into eight parts; 2 As and 6 Bs written on parts of the wheel. Emek divided his wheel into eight parts with four red and four white. Therefore, his representation of the sample space is not precise. Second, experiment design of his project is insufficient because the wheel in his project is incorrect. Experiment implementation, on the other hand, is successful because the wheel turns randomly, and results are recorded in variables properly. Multiple experiment simulations in Emek's project are successful: The user can conduct more than one experiment, see the result of each experiment cumulatively as variables, and reset the variables when needed. Lastly, his representation of outcomes is successful because results of the individual experiments are visible and variables are used properly. To conclude, Emek's project shows that he lacks a robust understanding of the sample space concept. Although his project is successful as a probability experiment, being able to define sample spaces is crucial to conduct fair probability experiments. His PAT pretest score is 4 and posttest score is 7. It may be concluded that the intervention helped Emek to develop knowledge of probability; however, he needs to further develop knowledge of sample spaces and basic events.

Almost half of the students' projects, in general, were not counted as successful in terms of user-game interaction (e.g., user input, feedback, contrasts in color of objects), and user-interface decoration. It seems many students neglected to consider every aspect of user-game interaction. It has to be noted that expecting students to produce a complete interface for the users of their

game with a comprehensive consideration of users' interaction needs is not fair because fifth and sixth graders do not have the knowledge of visual or interaction design.

Discussion and Conclusion

Analysis of students' projects as well as statistical tests on students' PAT scores showed that more than 90% of the students designed and implemented probability experiments in Scratch successfully. On basis of these findings, we can confirm that the intervention had a statistically significant positive effect on students' probability achievement. Hence, the results partially confirm the arguments that learning by video game design and computer programming help students to learn mathematical contents. It was observed that participants learned to reason with a problem abstractly and quantitatively when trying to model and coordinate a probability event in a game they designed. Among the coded game-development events, they were involved in imagining the objects of the game as they plan events, the interaction and sequence of diverse presentations, testing, and debugging. In testing and debugging their game strategies in Scratch codes, they refined their ideas to improve the design and clarified the connections with underlying concepts of probability. They also had opportunities for applying probability concepts and quantitative reasoning. In return, they learned to model a real-world problem using probability expressions.

Looking deeper into collected data, we also found that students' involvement in developing partial or complete Scratch projects about probability is critical. For example, students numbered 2, 16, and 21 had projects that were marked insufficient in terms of developing strategies for solutions of probability problems, but the difference between their probability achievement pretest and posttest scores were $+5$, $+2$, and $+6$, respectively. On the other hand, student numbered 6 had one project rated as creative and all projects of the students numbered 13, 14, and 22 were marked as successful, but the difference between their pretest and posttest scores were -1, 0, -3, and 0, respectively. Therefore, it can be inferred that even though a student cannot accomplish a particular probability task, she or he can still learn probability. This particular finding is in line with the findings of Wilensky (1996) and Abrahamson and Wilensky (2005) that students should be provided with opportunities to spend time on creating, modifying, and fixing their own models of probability.

Along with aforementioned encouraging results of this study, it is important to note that results also suggest that there is still room for improvement. Out of 14, students' average scores were 3.43 in pretest and 5.57 in the posttest. In other words, an average student still could not answer 8 out of 14 questions. Second, almost half of the students failed to develop algorithms for simulating multiple experiments at once in their projects. There might be various reasons behind these issues. For example, activities should have included more explicit

references to multiple simulations. In addition, diSessa (1997) asserts that a programming environment should have a low threshold and a high ceiling, that is, it should be comprehensible for beginners and helpful to relative experts. Scratch provides tools for beginners (Resnick et al., 2009). Conversely, the complexity of developing algorithms for multiple experiments in Scratch could have had a demotivating effect on students. To overcome this barrier, students should be given more time to master Scratch programming skills to exchange code blocks. Another temporary solution could be giving students prebaked functions or algorithms. Furthermore, it is also clear that students' learning of probability should have been measured with multiple methods. In similar studies, researchers used alternative methods to measure students' learning and understanding of probability concepts such as interviewing students (e.g., Wilensky, 1996), administering questionnaires (e.g., Fischbein & Schnarch, 1997), or analyzing students' discussions (e.g., Abrahamson & Wilensky, 2005; Hoemann & Ross, 1971).

Similar programming activities in a visual feedback-rich LOGO microworld helped middle school students learn transformational geometry concepts (Edwards, 1991) and angles and polygons (Clement & Battista, 1989). However, this study did not agree with some other studies conducted with LOGO and Scratch environments: For instance, Olive (1991) reported that ninth graders could not benefit from LOGO programming in learning geometry and recommended that LOGO activities should be connected to students' geometry experiences in earlier grades. Also, Taylor et al. (2010) did not observe fourth and fifth graders' learning of mathematics content, despite the fact that Scratch helped children to develop problem-solving strategies and reason systematically. Furthermore, Boyer (2010) reported that primary school students' development of multimedia artifacts with Scratch did not increase their knowledge of geometric solids, but the approach served as an alternative form of assessment. In this study, video game design and programming activities provided students with a learning setting in which they engaged in a meaningful context, constructed an understanding, and developed links between context and content of probability. In addition, the setting with visual feedback, language, and authentic experiences provided the students with opportunities for integrating content of probability and game ideas in their Scratch projects. Scratch-based game programming with instant feedback acted as an authoring tool for students to test their conceptual understanding of probability. Particularly, students as designers can instantiate the meaning of abstract concepts (such as value, variable, and probability expressions) and can test their reasoning (such as iterations, loops, and if conditionals) with different screen objects for a series of probability events. This interpretation maintains the view of constructionism on creating active learning through authoring and programming (Kafai et al., 1998; Ke, 2014).

The study indicates that students as game programmers experience a linked representation of probability concepts and procedures during their

program construction. The authoring environment supplied concrete events and object-based representation of probability concepts, thus supported students to develop connections between formal and informal conceptual understanding. Besides, the practice of planning, construction, development and debugging the Scratch scripts for video game composition enabled students to represent a process and reason out with a problem through probability concepts and expressions. In other words, video game programming has acted as a manipulative for the development of probabilistic thinking. The findings partially support the claim of recent research that computer game authoring may serve as a microworld or a learning setting for students to explore, depict, and test their domain knowledge (deHaan, 2011; Ke, 2014; McCue, 2011; Robertson & Howells, 2008).

Literature on teaching and learning probability indicates that students' ability to understand probability concepts increase over time (Memnun, 2008; Piaget & Inhelder, 1975; Shaughnessy, 1992). For instance, Piaget and Inhelder (1975) stated that students between ages 7 and 14 have no systematic approach to generating a list of possibilities, and that those students do not possess "the mathematical maturity to make an abstract model of a probability experiment" (Shaughnessy, 1992, p. 479). Additionally, Fischbein and Gazit (1984) reported that fifth grade students had more difficulty in learning probability than sixth and seventh grade students. In contrast, in this study, neither the quantitative nor the qualitative data collected showed a noticeable difference between fifth and sixth grade students, and it was found that students were successful in representing sample space of probability experiments in their projects. In game-programming as a context for creativity and design, students get invaluable chances to create and play with their mental representations of problems and solutions in the forms of video games (Akcaoglu & Koehler, 2014). In programming probability problems, students continuously debugged and developed codes to run the events of a game properly. All these processes were integral components of the engaging process of the game programming, which eventually resulted in improvements in students' understanding of probability concepts and processes.

This study provides evidence to support video game programming activities for learning basic probability concepts at the middle school level. It is concluded that video game programming is a good alternative for learners to build their own probabilistic models, run tests, analyze results, and understand fundamental structures of probability concepts. It is also an alternative to motivate students to spend more time with probability experiments.

Limitations and Future Work

Promising and encouraging results were obtained in increasing students' knowledge of probability through creative computing activities. The study, however,

had certain limitations. First, the study sample was not a random selection of students, but rather it was, in effect, a convenience sample. Second, the schools did not allow the researchers to collect data from a control group. Third, it was impossible to collect further qualitative data because we could not get parents to approve recording class videos and conducting in-depth interviews with students. Therefore, the reader must be cautioned against generalizing these results.

Findings of this research encourage further research in this field. First, it was found that programming video games with Scratch was effective for learning basic probability concepts, but students had difficulty in implementing multiple experiment simulations. A study should be conducted solely on multiple experiment simulations to find if the cause of this issue is due to the structure of the activities or the Scratch programming environment itself. If the latter is the case, developing extensions for Scratch to facilitate more complex algorithms and large amounts of data simulation can be considered as a future study. Besides studying the effectiveness of such activities in classroom settings, students' reflections on projects shared among themselves, and with others on online platforms, should be measured with both qualitative as well as quantitative methods and analyzed. Moreover, providing students with prebaked code blocks for probability functions or algorithms may be tested out in order to ease the understanding of multiple events of probability, and in turn, their effect on developing higher level of probabilistic thinking. Last but not least, a further study should examine developing virtual mechanisms for students to reflect their ideas, collect their individual reflections in a portfolio, and refer back to their past reflections whenever necessary.

Acknowledgments

First and foremost, we thank anonymous reviewers for their constructive comments. Second, we thank students who participated in the pilot and experimental studies, Koc Schools, and teachers Tumay Dovan, Eda Aydemir, and Meltem Onal for their support. Lastly, we thank Christina Pei for her help in preparing this manuscript.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

References

Abrahamson, D., & Wilensky, U. (2005). Problab goes to school: Design, teaching, and learning of probability with multi-agent interactive computer models. In D. Pratt, M.

B. Ottaviani & M. Meletiou (Eds.), *Proceedings of the 4th Conference of the European Society for Research in Mathematics Education* (pp. 570–580). Spain: Universitat Ramon Llull.

Abrahamson, D., Janusz, R., & Wilensky, U. (2006). There once was a 9-block: A middle school design for probability and statistics. *Journal of Statistics Education, 14*(2). Retrieved from http://www.amstat.org/publications/jse/v14n1/abrahamson.html

Akcaoglu, M., & Koehler, M. J. (2014). Cognitive outcomes from the game-design and learning (GDL) after-school program. *Computers & Education*, *75*(1), 72–81.

Aslan, U. (2014). *Fostering students' learning of probability through video game programming* (Unpublished MSc thesis). Boğaziçi University, Turkey.

Bar-On, E., & Or-Bach, R. (1988). Programming mathematics: A new approach in introducing probability to less able pupils. *International Journal of Mathematical Education in Science and Technology*, *19*(2), 281–297.

Baytak, A., & Land, S. M. (2011). An investigation of the artifacts and process of constructing computers games about environmental science in a fifth grade classroom. *Educational Technology Research and Development*, *59*, 765–782.

Biehler, R., & Pratt, D. (2012). Research on the reasoning, teaching and learning of probability and uncertainty. *ZDM*, *44*(7), 819–823.

Blau, I., Zuckerman, O., & Monroy-Hernandez, A. (2009). Children's participation in a media content creation community: Israeli learners in a scratch programming environment. In Y. Eshet-Alkalai, A. Caspi, S. Eden, N. Geri & Y. Yair (Eds.), *Proceedings of the Chais Conference on Instructional Technologies Research 2009: Learning in the Technological Era* (pp. 65–72). Israel: Open University of Israel.

Boyer, J. T. (2010). *Using Scratch for learner constructed multimedia: A design based research inquiry of constructionism in practice* (Doctoral dissertation). University of Florida, Florida.

Bransford, J. D., & Stein, B. S. (1984). *The ideal problem solver*. New York, NY: Freeman.

Brennan, K. (2011). *Creative computing: A design-based introduction to computational thinking*. Retrieved from http://scratched.gse.harvard.edu/sites/default/files/curriculumguide-v20110923.pdf

Brown, S. I., & Walter, M. I. (1983). *The art of problem posing*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Calder, N. (2010). Using scratch: An integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom*, *15*(4), 9–14.

Campbell, D. T., & Stanley, J. C. (1963). *Experimental and quasi-experimental designs for research*. Stamford, CT: Cengage Learning.

Clements, D. H., & Battista, M. T. (1989). Learning of geometric concepts in a LOGO environment. *Journal for Research in Mathematics Education*, *20*(5), 450–467.

Clements, D. H., & Sarama, J. (1997). Research on LOGO. *Computers in the Schools*, *14*(1), 9–46.

Cohen, J. (1990). Things I have learned. *American Psychologist*, *45*, 1304–1312.

Davies, C. M. (1965). Development of the probability concept in children. *Child Development*, *36*(3), 779–788.

deHaan, J. (2011). Teaching and learning English through digital game projects. *Digital Culture & Education*, *3*(1), 46–55.

Denner, J., Werner, L., & Ortiz, E. (2011). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts?. *Computers & Education*, *58*(1), 127–137.

diSessa, A. (1997). Twenty reasons why you should use boxer (instead of Logo). In M. Turcsanyi-Szabo (Ed.), *Learning & exploring with logo: Proceedings of the Sixth European LOGO Conference* (pp. 7–27). Budapest, Hungary: John von Neumann Computer Society.

Drier, H. S. (2000). The probability explorer: A research-based microworld to enhance children's intuitive understandings of chance and data. *Focus on Learning Problems in Mathematics*, *22*(3–4), 165–178.

Edwards, L. (1991). Children's learning in a computer microworld for transformation geometry. *Journal for Research in Mathematics Education*, *22*(2), 122–137.

Fischbein, E. (1975). *The intuitive sources of probabilistic thinking in children*. Dordrecht, The Netherlands: Reidel.

Fischbein, E., & Gazit, A. (1984). Does the teaching of probability improve probabilistic intuitions. *Educational Studies in Mathematics*, *15*(1), 1–24.

Fischbein, E., & Schnarch, D. (1997). The evolution with age of probabilistic, intuitively based misconceptions. *Journal for Research in Mathematics Education*, *28*(1), 96–105.

Goldberg, S. (1966). Probability judgments by preschool children: Task conditions and performance. *Child Development*, *37*, 157–167.

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments*, *1*(1), 1–32.

Hoemann, H. W., & Ross, B. M. (1971). Children's understanding of probability concepts. *Child Development*, *42*(1), 221–236.

Hoyles, C., & Noss, R. (1992). A pedagogy for mathematical microworlds. *Educational Studies in Mathematics*, *23*, 31–57.

Hoyles, C., & Sutherland, R. (1992). LOGO *mathematics in the classroom*. London, England: Routledge.

Hwang, G. J., Hung, C. M., & Chen, N. S. (2014). Improving learning achievements, motivations and problem-solving skills through a peer assessment-based game development approach. *Educational Technology Research and Development*, *62*(2), 129–145.

Kafai, Y., Ching, C. C., & Marshall, S. (1997). Children as designers of multimedia software. *Computers Education*, *29*(2), 117–126.

Kafai, Y., Franke, M. L., Ching, C. C., & Shih, J. C. (1998). Game design as an interactive learning environment for fostering students' and teachers' mathematical inquiry. *International Journal of Computers for Mathematical Learning*, *3*, 149–184.

Kafai, Y., Peppler, K., & Chiu, G. (2007). High tech programmers in low-income communities: Creating a computer culture in a community technology center. In C. Steinfield, B. T. Pentland, M. Ackerman & N. Contractor (Eds.), *Proceedings of the third communities and technologies conference* (pp. 1–19). New York, NY: Springer.

Kavasoglu, B. E. (2010). *The effects of game based teaching of probability on 6th–8th graders' mathematics achievement* (Unpublished Master's thesis). Gazi University, Turkey.

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.

Konold, C. (1995). Confessions of a coin flipper and would-be instructor. *The American Statistician*, *49*(2), 203–209.

Lave, J., Murtaugh, M., & de la Rocha, O. (1984). The dialect of arithmetic in grocery shopping. In B. Rogoff & J. Lave (Eds.), *Everyday cognition* (pp. 67–94). London, England: Cambridge University Press.

Lewis, C., & Sarah, N. (2012). Building upon and enriching grade four mathematics standards with programming curriculum. *Proceedings of the ACM Special Interest Group on Computer Science Education*, NY, USA, 57–62.

Maloney, J., Peppler, K., Kafai, Y. B., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth learning programming with Scratch. *Proceedings of the 39th ACM Technical Symposium on Computer Science Education*, Portland, USA, 367–371.

McCue, C. M. (2011). *Learning middle school mathematics through student designed and constructed video games* (Doctoral dissertation). University of Nevada, Reno, NV.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. M. (2010). Learning computer science concepts with Scratch. *Proceedings of the Sixth International Workshop on Computing Education Research*, Aarhus, Denmark, pp. 69–76.

Memnun, D. S. (2008). Difficulties of learning probability concepts, the reasons why these concepts cannot be learned and suggestions for solution. *Inonu University Faculty of Education Journal*, *9*(15), 89–101.

Mor, Y., & Sendova, E. (2003). Toontalking about mathematics. In I. Derzhanski, N. Dimitrova, S. Grozdev & E. Sendova (Eds.), *Proceedings of the International Congress MASSEE 2003* (pp. 36–43). Bulgaria: Borovets.

Olive, J. (1991). LOGO programming and geometric understanding: An in-depth study. *Journal for Research in Mathematics Education*, *22*(2), 90–111.

Papert, S., & Harel, I. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.), *Constructionism*. New York, NY: Ablex Publishing Corporation.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Sussex, England: Basic Books.

Papert, S., & Solomon, C. (1971). Twenty things to do with a computer. *Artificial Intelligence Memo*, *248*(3), 1–40.

Peppler, K., & Kafai, Y. (2007). What videogame making can teach us about literacy and learning: alternative pathways into participatory culture. In B. Akira (Ed.), *Proceedings of Digital Games Research Association 2007 Conference* (pp. 369–376). Tokyo, Japan: The University of Tokyo.

Piaget, J., & Inhelder, B. (1975). *The origin of idea of chance in children*. New York, NY: Routledge.

Polya, G. (1954). *How to solve it*. Princeton, NJ: Princeton University Press.

Resnick, M. (1996). New paradigms for computing, new paradigms for thinking. In Y. B. Kafai & M. Resnick (Eds.), *Constructionism in practice: Designing, thinking, and learning in a digital world* (pp. 255–268). Hillsdale, NJ: Lawrence Erlbaum Associates.

Resnick, M. (1997). *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. Cambridge, MA: MIT Press.

Resnick, M. (1998). Technologies for lifelong kindergarten. *Educational Technology Research and Development*, *46*(4), 43–55.

Resnick, M. (2006). Computer as paintbrush: Technology, play, and the creative society. In D. Singer, R. Golinkoff & K. Hirsh-Pasek (Eds.), *Play = Learning: How play motivates and enhances children's cognitive and social-emotional growth*. New York: NY: Oxford University Press.

Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *Proceedings of 6th ACM SIGCHI Conference on Creativity & Cognition,* Washington, DC, USA, 1–7.

Resnick, M. (2012). Reviving Papert's dream. *Educational Technology*, *52*(4), 41–46.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67.

Resnick, M., & Wilensky, U. (1998). Diving into complexity: Developing probabilistic decentralized thinking through role-playing activities. *Journal of the Learning Sciences*, *7*(2), 153–172.

Robertson, J., & Howells, C. (2008). Computer game design: Opportunities for successful learning. *Computers & Education*, *50*, 559–578.

Rosenbaum, E. (2009). Jots: Reflective learning in Scratch. *Proceedings of the 8th International Conference on Interaction Design and Children*, Como, Italy, 284–285.

Schoenfeld, A. H. (1985). Metacognitive and epistemological issues in mathematical understanding. In E. A. Silver (Ed.), *Teaching and learning mathematical problem solving*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Shaughnessy, M. (1992). Research in probability and statistics: Reflections and directions. In D. Grouws (Ed.), *Handbook of research on mathematics teaching and learning* (pp. 465–494). New York, NY: Macmillan Publishing Company.

Taylor, M., Harlow, A., & Forret, M. (2010). Using a computer programming environment and an interactive whiteboard to investigate some mathematical thinking. *Procedia Social and Behavioral Sciences*, *8*, 561–570.

Wilensky, U. (1996). Making sense of probability through paradox and programming: A case study in a connected mathematics framework. In Y. B. Kafai & M. Resnick (Eds.), *Constructionism in practice: Designing, thinking, and learning in a digital world*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Wilensky, U. (1997). What is normal anyway?: Therapy for epistemological anxiety. *Educational Studies in Mathematics*, *33*(2), 171–202.

Wolz, U., Stone, M., Pulimood, S. M., & Pearson, K. (2010). Computational thinking via interactive journalism in middle school. *Proceedings of 41st ACM Technical Symposium on Computer Science Education*, Milwaukee, USA, 239–243.

Author Biographies

Akpinar received his BSc degree in the field of measurement and evaluation in education. He then completed both his master's and PhD in the field of computer-based mathematics and science learning at Leeds University, UK. He has been with Boğaziçi University, Faculty of Education, since 1995, presently

chairing the Department of Computer Education and Educational Technology. His research interests and publications are in ICT education, computer-based learning/training, design of interactive learning environments, authoring systems, learning object design, use of technology for mobile, open, and e-learning.

Aslan received his BSc degree in the field of computer education and educational technology. He then completed his master's in the field of teaching mathematics and science. He conducts research studies on learning and teaching through computational methods. He is a software developer and founder of Odyssey R&D. He is currently working on his PhD at Northwestern University.