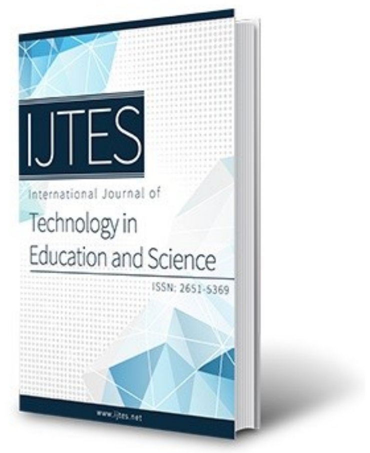

www.ijtes.net

# An Action Research Study on the Development of Object-Oriented Programming Course

**Osman Gazi Yildirim**
National Defense University, Turkey

**Nesrin Ozdener**
Marmara University, Turkey

**To cite this article:**

Yildirim, O. G. & Ozdener, N. (2021). An action research study on the development of object-oriented programming course. *International Journal of Technology in Education and Science (IJTES), 5*(4), 620-628. https://doi.org/10.46328/ijtes.277

IJTES
International Journal of Technology in Education and Science
2021, Vol. 5, No. 4, 620-628 https://doi.org/10.46328/ijtes.277

# An Action Research Study on the Development of Object-Oriented Programming Course

**Osman Gazi Yildirim, Nesrin Ozdener**

**Article Info**

***Article History***

Received:
30 March 2021
Accepted:
15 September 2021

***Keywords***

Action research
Game programming
Programming education
Object-oriented programming course

**Abstract**

Computer games are effective instructional tools used in programming courses to increase students' motivation and engagement. This participatory action research aims to redesign the Object-Oriented Programming course in which the first author is both the instructor and researcher to make it more effective and efficient. In the first step of the action research, data were collected for the definition and solution of the problem through questionnaires and semi-structured interviews. After this step, an action plan was created, and the Object-Oriented Programming course was redesigned as part of the action plan. In line with the objectives of the action plan, The Karting Microgame Template, a game prototype prepared by Unity 3D, was integrated into the course, and students were expected to add various game components (bonus collection system, a scoring system, collision mechanisms, etc.) to this game prototype using C # programming language. After the action plan creation phase, an action plan was implemented. The implementation phase was conducted in the 2019-2020 spring semester with 29 post-secondary students enrolling in Computer Technology Department at a vocational college in Turkey. After the implementation phase, data were collected through the Object-Oriented Programming achievement test, student and researcher diaries, and focus group interviews on measuring this implementation's effectiveness. This paper describes the difficulties encountered during the study, the students' views on this implementation, and the researchers' experiences in this process.

## Introduction

Computer programming competence is one of the skills that are perceived as vital in today's information society. Considering that software is used in our lives, providing effective programming education is critical in creating sectoral solutions and providing the workforce to work in this field. Similarly, having competence in object-oriented programming for students is vital because many experts practice it in the software development process (Brinda, Kramer, Hubwieser, & Ruf, 2015; Klump, 2001). However, object-oriented programming is a field that is generally considered difficult by students because it requires a high-level set of cognitive skills like problem-solving, abstraction, critical and mathematical thinking, testing, and debugging (Bonar & Soloway, 1983; Korkmaz & Altun, 2014; Lau & Yuen, 2009; Milne & Rowe 2002; Nevalainen & Sajaniemi, 2005; Seng, Yatim, & Hoe, 2018; Thota & Whitfield, 2010). According to Özdener (2008), even if students take many programming courses in high school and university education, students have difficulty in mastering programming skills and have misconceptions. With this regard, content and the required competencies affect students' attitudes and motivations towards programming courses. According to Balla & Király (2020), students do not prefer to work as programmers, and there is a lack of qualified workforce to work in computing. The study results conducted by Bergin & Reilly (2005) showed that 35% of the students had taken the programming course because it was compulsory, while 40% have taken the course for their career. The proportion of students who have taken the programming course due to their interests did not exceed 22%. Retention of interest in the field is becoming an urgent concern (Crenshaw, Chambers & Metcalf, 2008; Kurkovsky, 2013).

One of the most vital factors that play a role in students' choices for programming-related disciplines is their attitude and motivation towards programming (Alaoutinen & Smolander, 2010). However, it is challenging to motivate students decently (Seralidou & Douligeris, 2021). According to Kurkovsky (2013), many students prefer not to proceed with their career in computer science because they believe that the course assignments, projects, and materials have little pertinence to real life. In addition, students feel that their professional life will involve too much coding and that they cannot use their creative skills in their careers. Similarly, Beaubouef & Mason (2005) indicates that students often become disillusioned in computer science because of boring projects that lack connection with real-world applications. Mihci & Ozdener Donmez (2017) state that students desire a programming education curriculum that will provide them with an advantage in the future and allow them to work in different fields even if they receive teacher education. Donovan, Cotter, & O'Reilly (2020) cite "poorly delivered courses" may be one reason for the difficulties faced by students studying computer science.

Regarding students losing their motivation in programming lessons, claim that perception of course content as "boring" and lack of engaging instructional methods causes a decrease in motivation for programming courses (Attane & Kanjug, 2020; Ibrahim, Yusoff, Mohamed-Omar, & Jaafar, 2011; Song, Wang, Yang, & Zhang, 2020). Similarly, Petrovskaya (2019) asserts that low perceived competence due to subject difficulty and poor experience leading to negative attitudes are the main barriers towards motivation for programming. According to Vahldick, Mendes, & Marcelino (2014), learning to program requires plenty of practice and engagement; however, students can lack the needed motivation to engage with this need.

There is an ongoing discussion in the literature about "how to attract and engage students in programming courses" due to the students having too much difficulty in programming lessons and losing their interest in the courses over time (Coull & Duncan, 2011).  A variety of approaches to overcome this problem have been mentioned in the literature. One of the strategies proposed by a growing number of researchers is to reinforce students' motivation and improve attitudes by incorporating games and game programming into programming courses (Çelik, 2020; Kazimoglu, Kiernan, Bacon & Mackinnon, 2012; Yan, 2009). The rationale for this idea is to use a familiar and engaging context for students, allowing them to learn new concepts and arouse their curiosity through challenging them (Leutenegger & Edgington, 2007; Malliarakis, Satratzemi, & Xinogalos; 2014; Spires, Rowe, Mott, & Lester; 2011; Vassilev, 2015).

## Related Literature

Many scholars attempt to increase students' interests and course outcomes by tailoring games into course curriculums (Kucher, 2021). In this respect, the idea of using games in programming education is not a new practice (Vassilev, 2015). "Karel the Robot" (Pattis, Roberts, & Stehlik, 1995) is a pioneering programming microworld for teaching programming. Similarly, Robocode was developed to support teaching about java programming (Kazimoglu et al., 2012). Scratch (Maloney et al., 2004), ALICE (Cooper, Dann, & Pausch, 2003), code.org, CodeCombat, Greenfoot are also utilized to engage novice programmers to learn programming through creating games and practicing coding exercises (Mullins & Conlon, 2008; Seng et al., 2018; Wong, Hayati, & Tan, 2016). Zhu et al. (2019) mention that recent examples of game-creation environments (e.g., Cargo-Bot, Check iO, CodeSpells, etc.) are created to fuel the interest of students in programming, and these game-like environments comprise abstractions of several programming languages (programming languages, visual and text blocks) for particular programming competencies (comprehension, debugging, etc.) in various game genres (puzzle, etc.).

There are various studies in the literature on the use of game programming in programming education. Becker (2001) employed examples and assignments of game programming for increasing student motivation and engagement in introductory programming courses. Similarly, Leutenegger and Edgington (2007) incorporated a game-oriented approach to teaching an introductory programming course. Their results revealed that higher retention of basic programming concepts was achieved. Chen and Cheng (2007) conducted a study in which the students were expected to develop a computer game framework resulting in a higher feeling of accomplishment among students with enhanced programming skills. Long (2007) tested the effect of the Robocode game in teaching programming and found that Robocode improved the participant's motivation and achieved better effectiveness of course outcomes. Kurkovsky (2013) tailored mobile game development to an introductory computing course, causing improved students' engagement with the course material. Ouahbi et al. (2015) conducted an experimental study using Scratch environment with 69 high school students. The study results indicated that using Scratch increased students' motivation and empowered them to practice their programming studies. Similarly, Topalli and Cagiltay (2018) conducted experimental research to enrich their introductory programming course by integrating Scratch. The results showed that students' performance improved significantly. Begel et al. (2021) experimented with game programming on autistic first-year undergraduate students' communication self-efficacy. Researchers designed a code camp to teach students video game development using a block-based programming environment called MakeCode Arcade. Results showed that students were highly actuated to develop their games. In addition, their communication skills were improved by this intervention.

As shown in the literature, computer games can be effective tools for increasing the intrinsic motivation of CS students in teaching complex topics like object-oriented programming. Computer games can create engaging and motivational learning experiences when designed with clear learning objectives (Papadakis, 2018). A significant amount of existing research notes that computer games can be successfully integrated into programming lessons to facilitate recruitment and active engagement (Barnes et al., 2007; Fotaris, Mastoras, Leinfellner, & Rosunally, 2016). Research also asserts that most students prefer programming projects comprising computer games (Cliburn & Miller, 2008), and they learn and understand OO better (Chen & Cheng, 2007).

For the first author, the frustrations of reinforcing students' motivation in an OOP course provided the impetus to improve his current OOP course. This article traces the attempt to redesign an object-oriented programming course and adopt game programming into it. The main goal of the effort described here is to improve student success, motivation and satisfaction and investigate whether students possess concepts such as "enthusiastic, focused, and engaged" proposed by Garris, Ahlers, & Driskell (2002) through the redesigned OOP course. Furthermore, this paper aims to extend the literature on integrating game-based supplemental material for the OOP course.

## Method

### Research Design

This study was planned and implemented as action research. Action research steps adapted from Ferrance (2000) and Seberová and Malčík (2014) were followed in the study design (see Figure 1). In the first stage of the action research, the problems experienced by students and teachers in the current Object-Oriented Programming course were determined. To achieve this, data were collected through questionnaires and semi-structured interview forms. The modifications for the current Object-Oriented Programming course were recommended after the analysis of the collected data. After this stage, an action plan was created and implemented. This action plan, which was implemented for seven weeks, was analyzed, and the effectiveness of the implemented action plan was evaluated.

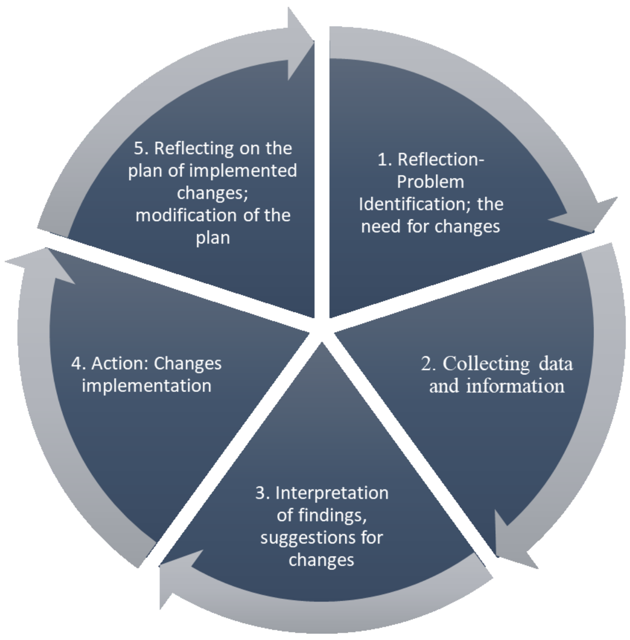
1. Reflection-Problem Identification; the need for changes
2. Collecting data and information
3. Interpretation of findings, suggestions for changes
4. Action: Changes implementation
5. Reflecting on the plan of implemented changes; modification of the plan

Figure 1. Action Research Process

*General Characteristics of Current Object-Oriented Programming Course*

The current Object-Oriented Programming course is three hours per week, including one theoretical and two hours of practice. Within the course objectives and the curriculum, students learn the fundamentals of object-oriented programming, classes, objects, class methods, interfaces, polymorphism, inheritance, etc. Some sample screenshots of object-oriented projects created during the implementation hours of the course are provided in Figures 2a, 2b, 2c, and 2d for giving information about in-course projects.

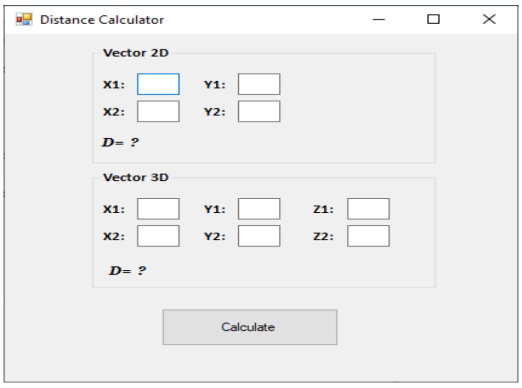

Figure 2a. A Project for Estimating Distance Between Points

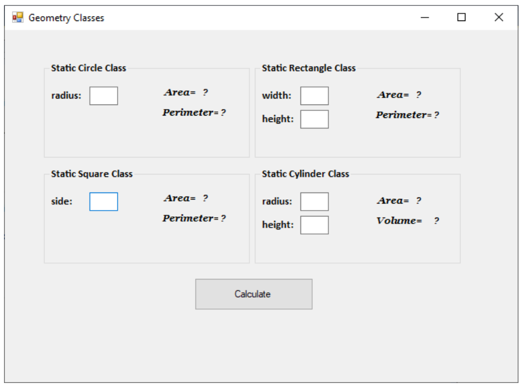

Figure 2b. A Project for Calculating Properties of Some Geometric Shapes Using Static Classes

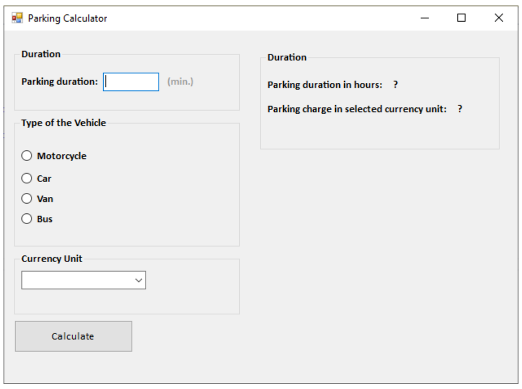

Figure 2c. A Project for Calculating Parking Charge of Different Types of Vehicles Using Inheritance and Polymorphism

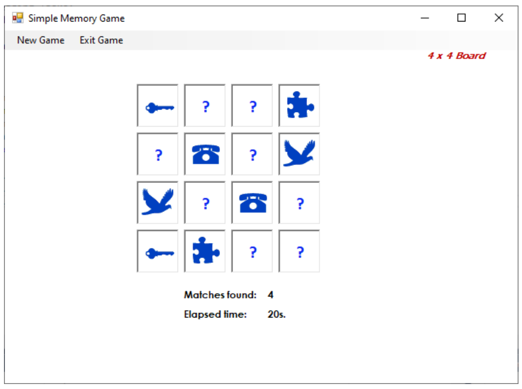

Figure 2d. A Simple Memory Board Game

As can be seen in the figures above, students develop form applications in the current course. For example, students create Vector 2D and Vector 3D classes that calculate distance between two points in 2-dimensional and 3-dimensional space using interfaces, inheritance, and polymorphism (Figure 2a). In Figure 2b, static geometric classes were developed, and static class methods calculate properties (area, perimeter, volume, etc.) of related geometric shapes. In another example, Figure 2c, a parking charge calculation project was developed using inheritance and polymorphism. In Figure 2d, a simple memory game was created for different board sizes like 4x4, 6x6, and 8x8. Object-oriented programming paradigm using C# programming language was utilized throughout all projects during the course. Although the language of the projects mentioned above is Turkish, translated versions into English were presented within the scope of this article.

## Research Questions

The research questions were categorized as analysis and design, application and evaluation, by considering the steps of the action research process. In this research process, the following questions were answered (see Table 1).

Table 1. Research Questions in the Action Research Phases

| **Analysis & Design Stage** | |
|---|---|
| **Action Research Steps** | **Research Questions** |
| Step 1. Reflection-Problem Identification; the need for the changes<br>Step 2. Collecting Data and Information | 1. What are the problems faced by students and lecturers in the current Object-Oriented Programming course? |
| Step 3. Interpretation of findings, suggestions for changes | 2. What kinds of changes can be made in the Object-Oriented Programming course to mitigate existing problems? |
| Implementation Stage | |
| Step 4. Action-Changes implementation | 3. What are the problems faced by the participants and the researcher during the implementation of the action plan? |
| Evaluation Stage | |
| Step 5. Reflecting on the plan of implemented changes; modification of the plan | 4. Has there been any change in students' perceptions of knowledge and skills?<br>5. What kind of change occurred in the participants' programming knowledge and skill levels after implementing the action plan?<br>6. What are the opinions of the participants about the Object-Oriented Programming course developed? |

## Participants

In accordance with the research questions in the study, three different participant groups took part in the study (see Table 2).

Table 2. Participants Groups of Action Research

| Stage of the Action Research | Type of the Participants | N |
|---|---|---|
| Analysis & Design Stage | Graduate students who have taken the course | 59 |
| | Lecturers of the current course | 5 |
| Implementation Stage | Participants of the current implementation | 29 |

As summarized in Table 2, data were collected from 59 graduate students who had previously taken the course to determine the problems experienced by students in the current OOP course in the analysis phase of the study. In addition, data were collected from five lecturers to reveal the problems of lecturers in the current course. After the action plan of the study was developed, 29 university students studying in the Computer Technology department at a state university in Turkey students participated in the implementation phase of the study. According to the data collected from the personal information form, participants of the implementation phase had a particular experience in programming and the C # programming language. On the other hand, 23 (80%) participants did not know the subjects of object-oriented programming such as classes, objects, access modifiers, constructor methods, class methods, inheritance, interfaces, polymorphism, etc. In comparison, 6 (20%) students were found to have very little knowledge of these subjects. Also, it was revealed that participants did not have prior knowledge of the game development editor Unity 3D, which was being used in the implementation phase of the action plan explained later.

**Data Collection Tools and Data Analysis**

A wide variety of data collection tools were utilized following the research steps and research questions in this action research. The data collection tools are presented in Table 2. The researchers developed the data collection tools in Table 2, the questionnaires, semi-structured interview forms, student self-report form, achievement test, and Programming Anxiety Scale. In addition to tools presented in Table 3, a personal information tool was developed to gather information of participants related to demographics and programming prior knowledge.

Table 3. Data Collection Tools

| Research Questions | Data Collection Tools |
|---|---|
| 1. What are the problems faced by students and lecturers in the current Object-Oriented Programming course? | Questionnaire<br>Semi-Structured Interview Form |
| 2. What kinds of changes can be made in the Object-Oriented Programming course to mitigate existing problems? | Questionnaire<br>Semi-Structured Interview Form |

| Research Questions | Data Collection Tools |
|---|---|
| 3. What are the problems faced by the participants and the researcher during the implementation of the action plan? | Student Diaries<br>Researcher Dairy<br>Focus Group Interview |
| 4. Has there been any change in students' perceptions of knowledge and skills? | Student Self-Report Form |
| 5. What kind of change occurred in the participants' programming knowledge and skill levels after implementing the action plan? | Achievement Test<br>Learning Motivation in Computer Programming Courses Scale<br>Computer Programming Self-Efficacy Scale<br>Programming Anxiety Scale |
| 6. What are the opinions of the participants about the Object-Oriented Programming course developed? | Student Diaries<br>Researcher Dairy<br>Focus Group Interview |

*Achievement Test*

The achievement exam is an exam developed to measure students' object-oriented programming knowledge and skills. This exam consists of a theoretical and practical exam. While the theoretical exam includes open-ended questions related to the explanation of OOP concepts, code writing, and comprehension tasks, the practical part is an exam in which students perform specific tasks in a game prototype assigned to them. Examples of open-ended questions in the theoretical exam are "*What is inheritance and what is it used for*?"; "*What are the differences between normal and static classes*?" can be given. Tasks in the practical exam included creating new classes, class variables, and methods, defining access specifiers for the class, performing collision detection, and implementing an object collection mechanism. A screenshot of the game prototype used in the practical part of the achievement test is provided in Figure 3.

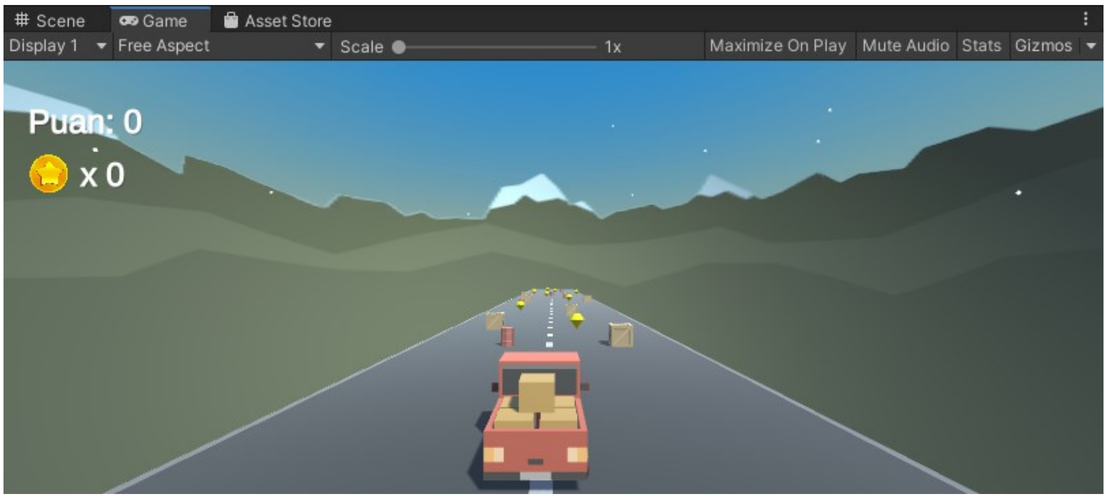

Figure 3. Screenshot of the Game Prototype Used in the Practical Exam

### *Programming Anxiety Scale*

Programming Anxiety Scale involves 14 statements on a 5-point Likert type scale ranging from 1 = never true to 5 = almost true. This scale consists of two subscales named "Significant Others" and "Programming Ability Anxiety." In this scale, while determining the participants' anxiety, the principle of "programming anxiety levels of the participants increases as the anxiety score increases" was chosen. The validity and reliability study of this scale was conducted with the participation of 392 university students.

### *Learning Motivation in Programming Courses Scale*

Learning Motivation in Programming Courses Scale (LMPCS) involves 19 statements on a 6-point Likert type scale. LMPCS is composed of six subscales: "Individual attitude and expectation," "Challenging goals," "Clear direction," "Reward and recognition," "Punishment," and "Social pressure and competition." The scale was developed by Law, Lee & Yu (2010) and adapted to Turkish by Avci & Ersoy (2018).

### *Computer Programming Self-Efficacy Scale*

Computer Programming Self-Efficacy Scale (CPSES) involves 31 statements on a 5-point Likert type scale. CPSES is a one-dimensional scale that was developed by Kukul, Gökçearslan, & Günbatar (2017). Although the original scale was developed for secondary school students, the researcher adopted the scale for university students with the participation of 343 university students before this study. In the data analysis process, descriptive statistics, Wilcoxon signed-rank test, and correlation analysis were conducted to analyze quantitative data. The significance level was set at .05 in all analyses. For qualitative data, content analysis was run, and codes and themes were created.

## Results

**Research Question 1. What are the problems faced by students and lecturers in the current Object-Oriented Programming course?**

To determine the problems experienced by the students in the current Object-Oriented Programming course, the researchers collected questionnaire data from 59 students who previously took the course. In addition, semi-structured interviews with eight students were conducted. As a result of the analysis of the collected data, problems were presented in five main categories (see Table 4).

Table 4. Problems of Students Regarding the Current OOP Course

| **Themes** | **Frequency (f)** |
|---|---|
| Course Content | |
| The course content is complex | 8 |
| The course content is uninteresting | 7 |
| Course content involves repetition of topics of previous programming | 5 |

| **Themes** | **Frequency (f)** |
|---|---|
| lessons | |
| The course content is not helpful for some | 5 |
| The Instruction of the Course | |
| The courses often include memorization | 10 |
| The pace of the course is fast | 6 |
| Students with different programming backgrounds are not taken into account | 5 |
| Examples in the lesson are not explained in detail | 4 |
| In-Course Projects | |
| Course projects are not related to daily life | 10 |
| Projects are not enjoyable | 8 |
| Several similar projects are created during the semester | 8 |
| Projects  are not visually attractive | 7 |
| Projects are not of good quality | 5 |
| Assessment Methods | |
| Extensive study-oriented for the exam | 8 |
| Projects and assignments have a low weight in the evaluation | 5 |
| Intervening Conditions | |
| Students have hardware and connectivity issues | 15 |
| There is not enough course material | 5 |

Regarding *the course content*, the participants stated that the course content could be tedious due to the repetition of the programming courses they had previously taken and that the topics were insufficient to attract their attention.

*"I don't think it adds anything to me, as I see things again that I already knew for me. It would be nice to learn new things." (AP2)*

*"I think it is helpful for the enthusiast to learn, but not very useful for me. " (AP15.)*

*"Those who have a certain foundation in the course can easily do things easily. But the course is complex in general, and people who do not have a basis in programming have difficulty understanding. " (AP27),*

Regarding the instruction of the course, it was revealed that participants who had not taken any programming course before had problems due to the pace of the course. In addition, some of the participants indicated that they had to learn the subject by memorizing.

*"I think the course content is processed very fast. While our friends, who were familiar with computer programming before, could do the lesson better, I understand it more difficult. It would be better if the lesson were explained on a simpler level." (AP3)*

*"I have a hard time learning what works where and what works. I had a hard time as we taught our lesson by heart and questioned everything that happens in me." (AP38),*

Regarding *the in-course projects*, the participants stated that they worked on the projects they developed during the implementation hours of the lesson on subjects that did not work for their daily life or did not interest them, which negatively affected their interest and motivation study.

> *"It can be* more productive and beneficial if the lessons are taught with a graphical emphasis to make the lessons more enjoyable and to attract attention, and if you go through examples that the student may like." (AP3)
>
> "There should *be a little more visual (appealing to the eye) and some little more professional applications."* (AP12)

Regarding *the assessment methods*, some of the participants stated that the examination system applied in the course guided them to memorization. They think that this situation prevents the learning of information and poses a threat to permanence.

> *"After the* exams pass, most of the information is forgotten."
>
> The programming course should not only consist of a visa or a final for the student."
>
> Regarding the intervening conditions, participants stated that they had problems with facilities. The basis of these problems is that students have a limited internet connection, do not have personal computers, and the course materials need to be improved.
>
> "It is very catchy when it is studied, but it becomes complicated to keep in mind because we do not have enough time and we do not have a computer of our own. "
>
> "I think the course is useful*, but learning to program in these possibilities becomes more difficult than it is."*

Besides the problems of students in the current OOP course, problems faced by the lecturers were investigated. For this reason, semi-structured interviews with five lecturers who had previously taught the course were carried out. As a result of the analysis of the collected data, two main categories were created (see Table 5).

Table 5. Problems of Lecturers Regarding the Current OOP Course

| Themes | Frequency (f) |
|---|---|
| Students' Attitudes towards the Course | |
| The motivation of some students towards the lesson is low | 5 |
| Some students do not regard course as beneficial | 4 |
| The course content and projects are not interesting for some students | 4 |
| Some students only study to pass the course | 4 |
| Incompetence of Students | |
| Some students' programming background is not sufficient | 3 |
| Some students have issues related to foreign language | 1 |

Instructors stated that some students had a low motivation or decreased over time because they believed they would not use programming in their professional lives. In addition, they noted that some students whose programming infrastructure was insufficient had difficulty understanding the lesson and lost their interest in the study. They pointed out that the projects carried out in the course were not enough to attract the attention of some students.

*"Generally, many students start to be interested in the lesson. However, after a few weeks, the students who can do the study maintain their motivation towards the class, and those who cannot do the lesson start to ask questions about what will work for them. Some students only work to pass the course. The only thing students are thinking about is how do I get through this lesson. Instead of asking how I run the program and complete it, they are busy getting through this lesson. " (AI2)*

*"For example, what application are we doing, a calculator. Students think that Windows already has a calculator, which they do not need to use a lot [calculator] anyway. In this case, some applications in the lesson do not attract the attention of our students. Because some of the practices we do are things that students will never use. " (AI3)*

*I observe that some students have difficulty understanding the lesson because they have foreign language problems. (AI1)*

**Research Question 2. What kinds of changes can be made in the Object-Oriented Programming course to mitigate existing problems?**

After analyzing the problem areas of the current course, the authors created solutions for problems in the design phase of the action research. Issues and suggestions for solutions are presented in Table 6. The recommendations were developed by utilizing an extensive literature review, the lecturers' offers, and students who have taken this course before.

Table 6. Problems and Recommendations for the Solutions

| Problem Sub-Category | Recommendations for the Solution of the Problem |
|---|---|
| Course Content | Enabling learning of new content<br>Including information and detailed content on the use of new technologies<br>Including the use of new trends, tools, and software |
| Instruction of the Course | Review of past topics in the course for students with the different programming knowledge level<br>Increasing teacher-student interaction<br>Creating opportunities for students to learn at their own pace<br>Tracking students who need help<br>More opportunities for peer learning<br>Using project-based activities in class<br>Taking into account individual differences |
| In-Course Projects | Including attractive, visually rich, enjoyable, and usable applications<br>Allowing the development of applications with difficulty levels that students with a good programming basis can benefit from<br>Including applications used in real life |
| Assessment Methods | Including in-class quizzes, assignments, and project design in the |

| Problem Sub-Category | Recommendations for the Solution of the Problem |
|---|---|
| | assessment |
| | Process-based assessment |
| Students' Attitudes towards the Course | Increasing the usability of the applications performed in the course |
| | Ensuring that students participate more in the lesson |
| | Following students closely in terms of homework, mini-exams, and projects and enabling reflective feedback |
| Intervening Conditions | Increasing the usage hours of computer classes |
| | Design of teaching materials for new course content |
| | Sharing the detailed explanations of the practices in the course |
| | Sharing of sample projects and details related to the lesson |
| | Sharing more course resources in the Turkish language |

As seen in Table 6, some suggestions were made for the problems of the current lesson. In line with these suggestions, the researchers redesigned the course by following Morrison, Ross, and Kemp's (2004) instructional design process. The teaching design of the researcher was carried out with Morrison, Ross, and Kemp's (2004) model for reasons such as including solutions to problems arising from teaching problems, focusing on the learner and context rather than looking at the instructional design in terms of content, having a flexible structure and not progressing linearly, and being convenient for teamwork. Within the scope of this instructional design model, the specific goals of the course were determined, characteristics of learners were identified, the OOP course content was clarified, instructional objectives and desired learning outcomes were defined, the appropriate mode of delivery was planned, and finally, the instruments for evaluation of learners were developed.

*Enriched Object-Oriented Programming Course*

An action plan including learning programming by game programming has been developed to increase the efficiency of the current Object-Oriented Programming course. The course content was redesigned in this context, and the topics and sub-topics presented in Table 7 were included in the course content.

Table 7. Enriched Course Content

| Week | Topic | Sub-Topics |
|---|---|---|
| 1 | Introduction to Unity 3D Editor | • Unity 3D Editor<br>• Transform Tools<br>• Component, Camera, Scene, Asset, Material, Prefab, and Rigidbody Concepts<br>• Adding C # Code to Objects<br>• Monobehavior Methods |
| 2 | Fundamentals of Object-Oriented | • Fundamentals of Object-Oriented |

| Week | Topic | Sub-Topics |
|---|---|---|
| | Programming | Programming |
| 3-4 | Introduction to Classes and Objects | • Class and Object Concepts<br>• Attribute and Behavior Concept<br>• Access Identifiers<br>• UML Class Diagrams<br>• Introduction of Classes and Objects in Unity 3D (Examination of Roll-A-Ball Game)<br>• Creating Classes (Points System and Bonus Classes) |
| 5-6 | Class Methods | • Creating Class Methods<br>• Using Class Methods of Other Classes |
| 7 | Static Classes | • Adding Unity 3D Methods to the Class<br>• Adding UI Elements to the Game<br>• Publishing the Game |

Within the scope of this action research, three hours of face-to-face lessons were taught each week. The first hour of these courses was carried out theoretically, and the second and third hours were practiced. As shown in Table 7, an introduction to Unity 3D editor was carried out in the first week. From the second week on, object-oriented programming topics have been covered. Students worked on the game prototype (Karting Microgame, a game prototype created by Unity) used as course material during the practice hours and added new components to the game prototype.

The initial and final version of the game prototype used in the lesson is presented in Figure 4a and Figure 4b. As seen in Figure 4b, the participants added new classes and functions to the game prototype; by adding a scoring system and the reward (bonus) collection mechanism. In addition, students redesigned the appearance of the game prototype. The research intervention was designed to underpin goal-oriented tasks to comprehend and utilize the object-oriented programming paradigm.

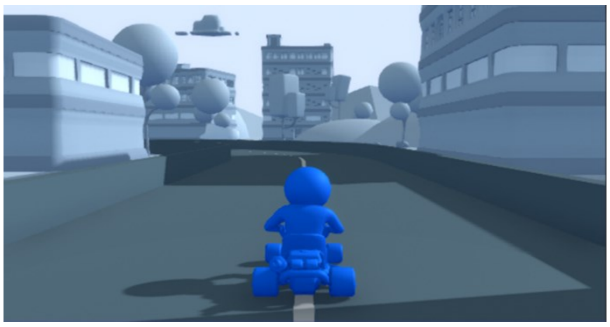
Figure 4a. Initial Version of the Game Prototype

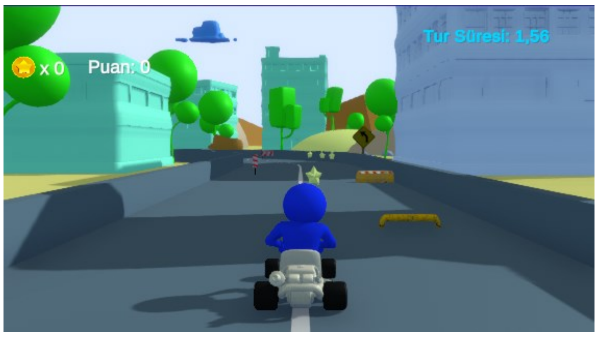

Figure 4b. Final Version of the Game Prototype

**Research Question 3. What are the problems faced by the participants and the researcher during the implementation of the action plan?**

The instructor recorded the problems encountered while implementing the action plan in a diary based on the students' statements in the lessons, the digital diaries, and observations of the students. During the implementation process, some action decisions were taken to solve these problems. The problems encountered during the implementation phase and the action decisions taken to solve the issues are presented in Table 8.

Table 8. Problems Arising During the Implementation of the Action Plan and Action Decisions Regarding the Solution of the Problems

| **Problems** | **Action Decisions for the Solution of Problems** |
|---|---|
| Due to time constraints, detailed information could not be shared about the introduction of the Unity 3D Editor | Video lectures and presentations have been added to the course portal about Unity 3D Editor |
| Student diaries were not filled in detail | Students were warned to fill in their diaries in detail |
| Two students missed classes in the first two weeks | Video lessons were created for each subject every week |
| Students often encountered errors when assigning values to class variables | Sample applications were made to debug errors encountered during the assignment of objects in the courses |
| It was observed that some students tried to memorize the codes instead of learning the subject. | Reflective questions were asked to the students to question for what purpose the class they created, which class variables and methods it had, what they would do, and their relationship with other classes<br>Students have been assigned to create UML diagrams with detailed descriptions of the classes to be added to the game prototype |
| Visual Studio Community 2015 software and Unity 3D version 2018.4.17f have repeatedly failed due to .NET Framework incompatibility. | Visual Studio Community 2017 software was installed in classrooms instead of Visual Studio Community 2015 |
| Students wanted to use different bonuses and assets, but the internet connection in the computer classroom is limited | Different assets were downloaded by the instructor and shared with the students on the course portal. |
| Some students reported that they couldn't complete their projects alone | Students were encouraged to get help from their friends during the lesson<br>Peer learning was supported through fostering group work |
| Some students had difficulty understanding the reasons for the bugs they encountered in their projects | One lesson was devoted to the most common mistakes, and applications on exception handling were carried out |

As seen in Table 8, some problems were experienced during the implementation of the action plan due to lack of time, lack of resources, software problems, and readiness levels of some students. Problems encountered were tried to be solved by obtaining expert opinions.

**Research Question 4. Has there been any change in students' perceptions of knowledge and skills?**

Students were asked to indicate their perceived programming knowledge and skill levels before and after implementing the action plan. An interview form was used for this. The change in the participants' programming knowledge and skill levels before and after the application is presented in Figure 5.

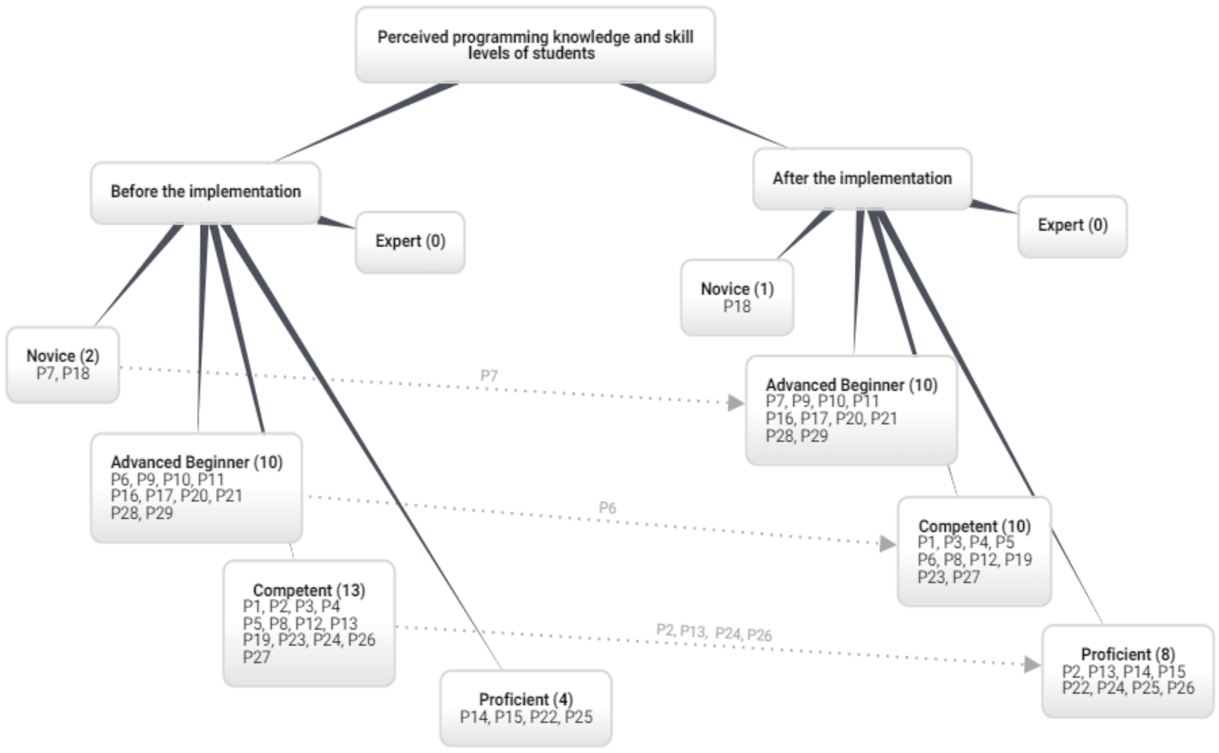
Perceived programming knowledge and skill levels of students
Before the implementation
Expert (0)
Novice (2) P7, P18
Advanced Beginner (10) P6, P9, P10, P11 P16, P17, P20, P21 P28, P29
Competent (13) P1, P2, P3, P4 P5, P8, P12, P13 P19, P23, P24, P26 P27
Proficient (4) P14, P15, P22, P25
After the implementation
Expert (0)
Novice (1) P18
Advanced Beginner (10) P7, P9, P10, P11 P16, P17, P20, P21 P28, P29
Competent (10) P1, P3, P4, P5 P6, P8, P12, P19 P23, P27
Proficient (8) P2, P13, P14, P15 P22, P24, P25, P26
P7
P6
P2, P13, P24, P26

Figure 5. The Change in Students' Perceived Programming Knowledge and Skill Levels before and after the Implementation

According to Figure 5, there was a change in the perceived programming knowledge and skill level of 6 (20.7%) participants. While 23 (79.3%) participants described themselves as an advanced beginner and competent level in terms of programming skills before the application, there were 20 (69%) participants after the implementation. In addition, while 17 (59%) participants described themselves as competent and proficient in programming skills before the implementation, there were 18 (62%) participants after the implementation. In other words, after the implementation, the number of participants who described themselves as an advanced beginner and competent decreased, and the number of participants who described themselves as competent and proficient increased. When the six participants who experienced a change in the perceived programming skill level were evaluated, the difference mainly was from competent to proficient (n = 4, 66.7%).

**Research Question 5. What kind of change occurred in the participants' programming knowledge and skill levels after implementing the action plan?**

Achievement test was utilized for examination of the difference between pretest and posttest statistically. The means of pretest-posttest scores, standard deviations, and differences are presented in Table 9. Before the comparison, it was checked whether test scores showed a normal distribution. According to the Shapiro-Wilk test results, the pretest scores of the participants were not normally distributed ($p <.05$), while the posttest scores were normally distributed ($p> .05$). For this reason, the Wilcoxon signs test was used to compare the pretest and posttest scores of students, as the Wilcoxon signs rank test was designed for evaluating participants in two different situations (Pallant, 2015). The Wilcoxon sequential signs test results performed for the object-oriented programming achievement test revealed that there were statistically significant changes in the success scores of the participants after participating in the pilot study with a large effect size ($r = .62$) ($z = -4.707$, $p <.001$). According to this result, object-oriented programming education significantly affected the students' object-oriented programming knowledge and skills.

Table 9. Pretest Posttest Means, Standard Deviations, and Difference

| | | Pretest | | Posttest | | |
|---|---|---|---|---|---|---|
| Variable Score | N | M | SD | M | SD | Difference |
| Achievement Test | 29 | 6.72 | 6.58 | 60.49 | 9.27 | 53.77 |

Posttest means and standard deviations of utilized scales are summarized in Table 10.

Table 10. Posttest Means, Standard Deviations, and Difference Values for Utilized Scales

| | | **Posttest** | |
|---|---|---|---|
| **Variable Score** | **N** | **M** | **SD** |
| Learning Motivation in Programming Courses Scale | | | |
| Individual attitude and expectation | 27 | 4.44 | .73 |
| Challenging goals | 27 | 3.27 | 1.33 |
| Clear direction | 27 | 4.02 | 1.09 |
| Reward and recognition | 27 | 4.20 | 1.23 |
| Punishment | 27 | 3.06 | .88 |
| Social pressure and competition | 27 | 2.60 | 1.34 |
| Total Score | 27 | 3.63 | .95 |
| Programming Anxiety Scale | | | |
| Significant Others | 27 | 2.37 | 1.06 |
| Programming Ability Anxiety | 27 | 2.87 | 1.06 |
| Total Score | 27 | 2.65 | 1 |
| Computer Programming Self-Efficacy Scale | | | |
| Total Score | 27 | 3.57 | .83 |

Correlation analysis was employed to examine the relationship between the participants' motivation towards the programming course, programming anxiety, self-efficacy perception towards programming, and achievement test posttest scores (see Table 11). The results of the Pearson correlation indicated that none of the sub-scales of motivation scale towards computer programming course and the participants' achievement test posttest scores correlate, ranging from r = −.28 to r = .07. Similarly, there was a non-significant correlation of .11 (p >.05) between participants' perceived self-efficacy towards programming and achievement test posttest scores. On the other hand, while Significant Others sub-scale of computer anxiety scale and achievement test posttest scores were significantly correlated, r = -.39, p < .01, Programming Ability Anxiety sub-scale was not significantly correlated, r = -.35, p > .05

Table 11. Correlation Analysis Results

| | Individual attitude and expectation | Challenging goals | Clear direction | Reward and recognition | Punishment | Social pressure and competition | Significant Others | Programming Ability Anxiety | Programming Self-Efficacy | Achievement test (posttest) |
|---|---|---|---|---|---|---|---|---|---|---|
| Individual attitude and expectation | — | | | | | | | | | |
| Challenging goals | .54** | — | | | | | | | | |
| Clear direction | .56** | .74** | — | | | | | | | |
| Reward and recognition | .57** | .30 | .40* | — | | | | | | |
| Punishment | .56** | .34 | .31 | .61** | — | | | | | |
| Social pressure and competition | .51** | .64** | .37 | .32 | .67** | — | | | | |
| Significant Others | -.07 | -.09 | -.38* | -.23 | .04 | .01 | — | | | |
| Programming Ability Anxiety | -.24 | -.37 | -.59** | -.32 | -.01 | -.02 | .76** | — | | |
| Programming Self-Efficacy | .50** | .34 | .50** | .48* | .22 | .15 | -.59** | -.68** | — | |
| Achievement test (posttest) | -.01 | .06 | .07 | .07 | -.28 | -.07 | -.39* | -.35 | .11 | — |

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

As Table 10 indicates, Significant Others (-.38) and Programming Ability Anxiety (-.59) sub-factor of computer programming anxiety scale were negatively associated with Clear direction sub-factor of motivation towards the programming course scale, p<.01 and p<.05, respectively. Furthermore, Significant Others (-.59) and Programming Ability Anxiety (-.68) sub-factor of the computer programming anxiety scale were negatively associated with participants programming self-efficacy perceptions, p<.05.

**Research Question 6. What are the opinions of the participants about the Object-Oriented Programming course developed?**

Determining the participants' opinions towards the intervention was carried out by analyzing the digital dairies of participants and the focus-group interview. Participants expressed their overall impressions about the course, intervention, and game prototype in the focus-group interview. Besides these data collection tools, the researcher dairy also analyzed to obtain the instructor's views on the effects of implementation.

*Student and Researcher Diary*

Students' digital dairies were investigated to determine the positive and negative perceptions of the participants about the course application, their thoughts about the use of the Unity 3D editor, and the implementation. Participants' positive perceptions of the lesson could be listed as enjoyable, helpful, motivating, and engaging. Participants expressed their positive perceptions about the course with the following sentences:

*"I think it is a more enjoyable lesson than regular lessons, especially by programming games, it will be much more fun to create. (IP12),*

"*Unity engine is more fun and instructive [than visual programming] versus visual programming." (IP2),*

*"This lesson attracted my attention more than other lessons; I think I will be interested in other lessons." (IP9),*

*"Learning programming by creating games is frankly more fun. It was good for raising awareness, it helped me in this way because the different attachments are in a curious and challenging way." (IP5),*

*"Today, we switched to Unity. It is beneficial and exciting in terms of visuality. The course made it more interesting to move to Unity, which is more vivid and visual than outdated and boring subjects." (IP3).*

*"I like it to continue this way. I like creating something out of nothing and using visuals." (IP15),*

*"I enjoyed this lesson. It is interesting to have a game in the lesson. But since it is only an introductory part, I had a little difficulty." (IP13),*

*"Visually, the method we use is more instructive and interesting than it is available." (IP8),*

*"I think the method we use in the lesson is more memorable, logical, and even more instructive. Form and consoles are boring, I think. It provides motivation and attracts attention". (IP27).*

The researcher also conveyed the positive observations he experienced about the participants during the lessons in the researcher's diary with the following sentences:

*“Today we started using the Unity 3D editor for the first time. Before I began to use it, I talked about a game, its types, and concepts such as the game engine concept—the scene, asset, component, texture,*

*material, etc. I talked about the basic concepts. I mentioned the Unity 3D editor windows (scene, game, hierarchy view, etc.). I think my students are interested. They ask me how to change the Unity skybox, collide objects, and rotate them faster. Their enthusiasm pleases me very much. I observe that their interest and motivation towards the lesson have increased. I try to answer almost all questions. I encourage them to keep small notes." (Researcher diary, 02.07.2020).*

Participants' negative perceptions of the lesson can be described as challenging and time-consuming. Participants expressed their negative perceptions about the course with the following sentences:

*"What did I not fully understand in this lesson? In general, I do not understand the lessons we teach during the day; I think I do when I repeat them later with the help of friends. I could not finish the car project today, so I could not understand anything. I plan to repeat and understand as soon as possible. I have to create time for this." (IP16),*

*"I'm doing class time, but I don't remember much because we can't deal with codes 24/7." (IP14),*

*"In this lesson, we made examples from the next level for the subject. However, in this lesson, I have forgotten some things about the topic and have to repeat them before getting more intense. In this lesson, I could not quite understand the examples. I think to repeat the subject as soon as possible and come to a better next week." (IP13),*

*"We are working on the sample game; trying to understand it is time-consuming. Modifying a game requires more work on Unity. " (A.D.), "I'm not very good with programming. We are working with my friend. He tells me the places I don't understand. Playing games is difficult for me." (IP11).*

The participants did not use the Unity 3D editor before the Object-Oriented Programming course. Participants initially found it complex as they first met Unity 3D in this lesson. The opinions of the participants on this subject are as follows:

*"We have started to use Unity. It was a little difficult to understand because it was a program I saw for the first time." (IP20),*

*"I am using Unity for the first time. It seemed confused to me. The sample game in the lesson is also difficult for someone who has never used Unity. When I add something to the game, I'm afraid it breaks down." (IP23),*

*"The most difficult thing this week was to learn the content and menus of the program, namely the application, and forget the places quickly." (IP18),*

*"I could not fully understand the operations such as right-click navigation, WASD keys, changing the views of the XYZ axes, I will ask again." (IP17).*

The instructor conveyed the problems he observed in the students during the lesson in his diary as follows:

*"I observe that some students try to memorize the topics. When I ask questions to them, I see that concepts such as class, object, class-component relation, class method do not fit well for some. Some students ask their friends and me for help very often. I made it possible for them to get help from each other while completing their projects to learn the topics better. Some students work with their friends and benefit from their expressions." (Researcher diary, 23.07.2020).*

*Focus-Group Interview*

The students stated that game programming was a very demanding and complicated subject in the focus group discussion. However, they indicated that game development was an enjoyable and instructive process. The majority of the students stated that they preferred programming learning by programming games over the traditional method. The students noted that it would be advantageous for them to know more about the editor before working on the prototype. They stated that the subject of introducing the Unity 3D editor should be more than a week. In addition, they emphasized that the course should include more weeks as the number of weeks. The participant named IP3 expressed his thoughts about the duration of the lesson with the following sentences:

> *"To create a program, it is necessary to look for ideas. On the other hand, we got better results by improving the application of the current program and getting less tired in less time. I think the time interval allotted to the lesson should be longer. In this way, we could focus more on theoretical issues. Because house construction cannot be started directly, it is necessary to learn how to lay bricks and mix mortar first. We knew this by doing it on the application that existed in a limited time. "*

Finally, when the participants were asked to evaluate the game prototype used as course material, the participants stated that the prototype was moderately complex and that different features could be added. In this way, they mentioned that the quality of the game could increase visually and functionally. When asked what these properties could be, the participants; (i) different tracks or different vehicles (race cars with different models, race engines, etc.) could be selected at the beginning of the game, (ii) the game could be multiplayer, (iii) visual effects to the game (sparkle when collecting bonuses, etc.) (see Figure 6). In addition, participants indicated that different game genres (first-person shooter, role-playing, virtual-reality, etc.) could be developed.

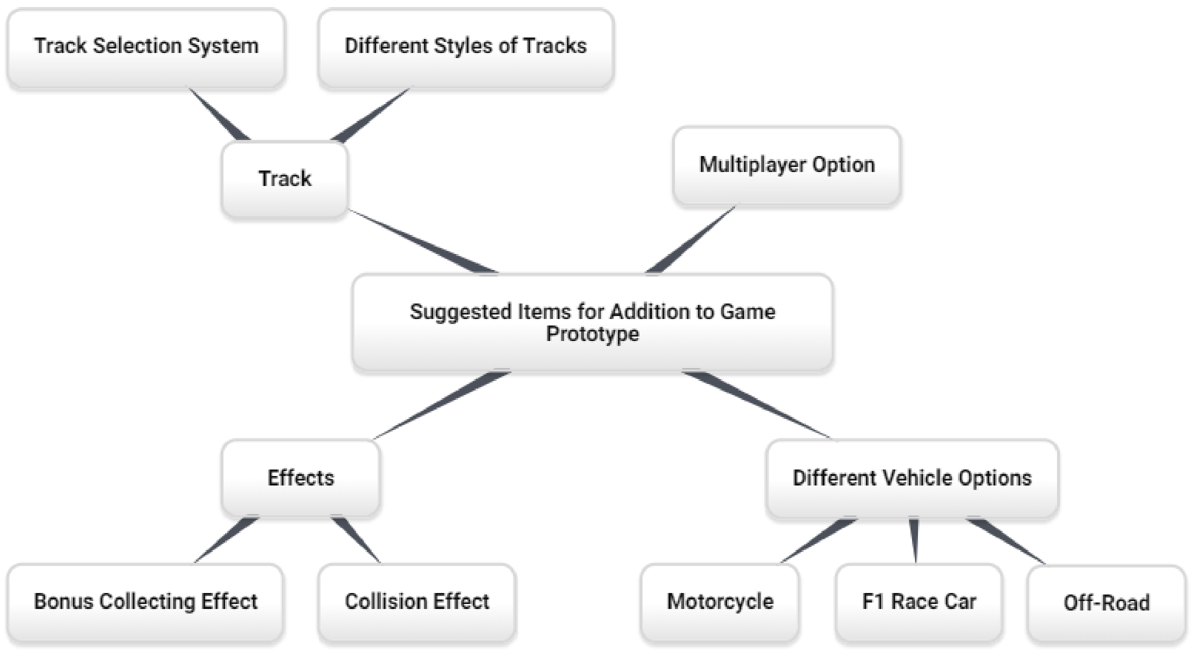

Figure 6. Suggestions for Game Prototype

## Discussion

The research findings indicated that learning to program in an object-oriented manner using games is an effective learning approach. Learning OOP within a game design context piques students' interest and curiosity and ensures students' active engagement because the vast majority of the students are interested in playing digital games. This result extends to those by other researchers studying game-based learning (Javidi & Sheybani, 2014; Ladislav & Stoffová, 2019; Soares, Martin, & Fonseca, 2015; Theodoraki & Xinogalos, 2014). In addition, it was found that most of the students were highly motivated to develop their games. This result is also consistent with findings of related literature (Kurkovsky, 2013, Begel et al., 2021, Topalli & Cagiltay, 2018). The results of the current study assert that it is critical to engage students with motivating tasks that interest them because, as Hmelo-Silver mentions, "intrinsic motivation occurs when learners work on a task motivated by their interests, challenges, or sense of satisfaction" (Hmelo-Silver, 2004, p. 241). Consistent with previous work (Chen & Cheng, 2007; Montes, Hijón-Neira, Pérez-Marìn, & Montes, 2021), a sense of accomplishment was stimulated, and the perceived programming knowledge level of some students increased. However, designing and utilizing a playable computer game from scratch can be challenging (Kurkovsky, 2013; Ladislav & Stoffová, 2019). The current study results showed that using a prepared application framework (Chen & Cheng, 2007) can be a profound solution for this problem, and modifying a game prototype could be an engaging and motivating tool.

Another result of the current study is related to the evaluation of the course outcomes. Achievement test scores significantly increased after the implementation of the action. The explanation of this result can be expressed by the fact that many objects and objects interact in the games (Chen & Cheng, 2007). Because modifying a game is a time-consuming process, as students' diaries confirmed, OOP skills might be broadened by so much time spent in adding features to one game prototype. On the other hand, several concepts seemed challenging for some students, and they could not complete their projects without having peer support.

In this study, correlation analysis was also utilized to investigate the association between the participants' motivation towards the programming course, programming anxiety, self-efficacy perception towards programming, and achievement test posttest scores. Surprisingly, the results of the Pearson correlation indicated that none of the sub-scales of motivation scale towards computer programming course and the participants' achievement test posttest scores correlated. Similarly, there was a non-significant correlation between participants' perceived self-efficacy towards programming and achievement test posttest scores. This finding contradicts previous research, which asserts a significant relationship between perceived self-efficacy and achievement (Law et al., 2010; Ramalingam, LaBelle, & Wiedenbeck, 2004; Yağcı, 2016).

On the other hand, consistent with the former findings (Connolly, Murphy, & Moore, 2007; Owolabi, Olanipekun, & Iwerima, 2014), one sub-scale of the programming anxiety scale was correlated with achievement test posttest scores. In addition, the Significant Others and Ability Anxiety sub-scales of the computer programming anxiety scale were negatively associated with participants' programming self-efficacy perceptions. Supporting the results of previous research (Nolan, Bergin, & Mooney, 2019), the current study results suggest that programming anxiety is an essential factor for programming achievement and programming self-efficacy perception.

Regarding the issues related to implementing the action plan, it was revealed that there were most problems with understanding class-object relationships, exception handling, software incompatibilities, and available possibilities. In addition, due to the pandemic, the content of the course was reduced. For this reason, the inheritance topic was not instructed, and the exception handling topic was taught when needed. As revealed in the student diaries and focus group interview, a week devoted to introducing the Unity 3D editor was insufficient for students to gain competence. It is recommended to allocate 2-3 weeks to master using the Unity game engine in this context.

## Conclusion

Object-oriented programming can be challenging for both students and instructors. Many studies in the literature (e.g., Kurkovsky, 2013; Seng et al., 2018; Zainal Abidin, Arsad, Muslim, & Masrom, 2020) try to find the best solution to this problem. This paper has reported action research incorporating modifying a game prototype to increase students' intrinsic motivation and course outcomes. Following the study's objectives, a process involving analysis, design, implementation, and evaluation steps was carried out. In conclusion, the results of current action research were positive regarding incorporating game development and modification for learning object-oriented programming. Students perceive learning programming by programming games as effective and beneficial. In addition, an increase in the motivation of students towards the object-oriented programming course was observed. The majority of students preferred game programming to traditional projects. However, it was also revealed that game programming was a complex subject and required much effort.

The results of this study are limited to the participants and context of the current research and should not be generalized to students in other contexts. It is crucial to validate the results in future research. Experimental studies, including many participants and different programming languages, different game prototypes, or different countries, could be conducted to draw more objective and reliable results. Furthermore, the change in programming motivation and anxiety in the implementation process was not examined. Studies that examine the difference in these factors can be conducted. Finally, studies that recommend a curriculum in a game-based OOP course can be carried out.

## Notes

The study was derived from the first author's ongoing dissertation.

## References

Alaoutinen, S., & Smolander, K. (2010, June). Student self-assessment in a programming course using bloom's revised taxonomy. In *Proceedings of the fifteenth annual conference on Innovation and technology in* *computer science education* (pp. 155-159).

Attane, P., & Kanjug, I. (2020, November). A Study of Learner's Mental Model and Motivation Using Constructivism Online Learning Environment to Promote Programming in Rural School. In *International Conference on Innovative Technologies and Learning* (pp. 361-366). Springer, Cham.

Avcı, Ü. & Ersoy, H. (2018). Bilgisayar Programlama Derslerinde Öğrenme Motivasyonu Ölçeğinin Türkçe Uyarlaması: Geçerlilik ve Güvenilirlik Çalışması. *Journal of Higher Education & Science/Yüksekögretim ve Bilim Dergisi, 8*(1).

Balla, T., & Király, S. (2020). A Discussion of Developing a Programming Education Portal. *Central-European Journal of New Technologies in Research, Education and Practice*, 1-14.

Barnes, T., Richter, H., Powell, E., Chaffin, A., & Godwin, A. (2007, June). Game2Learn: building CS1 learning games for retention. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 121-125).

Beaubouef, T., & Mason, J. (2005, June). Why the high attrition rate for computer science students: Some thoughts and observations. SIGCSE Bulletin, 37, 103–106

Becker, B. W. (2001, February). Teaching CS1 with Karel the Robot in Java. In *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education* (pp. 50-54).

Begel, A., Dominic, J., Phillis, C., Beeson, T., & Rodeghero, P. (2021, March). How a Remote Video Game Coding Camp Improved Autistic College Students' Self-Efficacy in Communication. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 142-148).

Bergin, S., & Reilly, R. (2005, February). Programming: factors that influence success. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education* (pp. 411-415).

Bonar, J., & Soloway, E. (1983). Uncovering principles of novice programming. *In Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages* (pp. 10-13).

Brinda, T., Kramer, M., Hubwieser, P., & Ruf, A. (2015). Towards a competency model for object-oriented programming. *In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 345-345).

Chen, W. K., & Cheng, Y. C. (2007). Teaching object-oriented programming laboratory with computer game programming. IEEE Transactions on Education, 50(3), 197-203.

Cliburn, D. C., & Miller, S. M. (2008). What makes a" good" game programming assignment?. *Journal of Computing Sciences in Colleges*, *23*(4), 201-207.

Connolly, C., Murphy, E., & Moore, S. (2007, September). Second chance learners, supporting adults learning computer programming. In *international conference on engineering education–ICEE*.

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. In *Proceedings of the 34th SIGCSE technical symposium on computer science education*, Reno, Navada, pp. 191–195

Coull, N. J., & Duncan, I. M. (2011). Emergent requirements for supporting introductory programming. *Innovation in Teaching and Learning in Information and Computer Sciences*, *10*(1), 78-85.

Crenshaw, T. L., Chambers, E. W., & Metcalf, H. (2008, March). A case study of retention practices at the University of Illinois at Urbana-Champaign. In *Proceedings of the 39th SIGCSE technical symposium on*

*Computer science education* (pp. 412-416).

Çelik, H. C. (2020). The effect of modeling, collaborative and game-based learning on the geometry success of third-grade students. *Education and Information Technologies, 450*(25), 449–469.

Donovan, R., Cotter, J., & O'Reilly, R. (2020, June). Improving Academic Performance Amongst First Years Computer Science Students Through Goal-Setting. In *2020 31st Irish Signals and Systems Conference (ISSC)* (pp. 1-6). IEEE.

Ferrance, E. (2000). *Action research*. LAB, Northeast and Island Regional Education Laboratory at Brown University.

Fotaris, P., Mastoras, T., Leinfellner, R., & Rosunally, Y. (2016). Climbing up the Leaderboard: An Empirical Study of Applying Gamification Techniques to a Computer Programming Class. *Electronic Journal of e-learning*, *14*(2), 94-110.

Garris, R., Ahlers, R., & Driskell, J.E. (2002). Games, motivation, and learning: A research and practice model. *Simulation& Gaming, 33*(4), 441-467.

Hmelo-Silver, C. E. (2004). Problem-based learning: What and how do students learn?. *Educational psychology review, 16*(3), 235-266.

Ibrahim, R., Yusoff, R. C. M., Mohamed-Omar, H., & Jaafar, A. (2011). Students' perceptions of using educational games to learn introductory programming. *Computer and Information Science, 4*(1), 205.

Javidi, G., & Sheybani, E. (2014). Teaching computer programming through game design: A game-first approach. *GSTF Journal on Computing (JoC), 4*(1), 17-22.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences, 47*, 1991-1999.

Klump, R. (2001). Understanding object-oriented programming concepts. *In Power Engineering Society Summer Meeting IEEE, 2*, 1070-1074.

Korkmaz, Ö., & Altun, H. (2014). Adapting computer programming self-efficacy scale and engineering students' self-efficacy perceptions. *Participatory Educational Research, 1(1)*, 20-31.

Kucher, T. (2021). Principles and best practices of designing digital game-based learning environments. *International Journal of Technology in Education and Science*, *5*(2), 213-223.

Kukul, V., Gökçearslan, Ş., & Günbatar, M. S. (2017). Computer programming self-efficacy scale (CPSES) for secondary school students: Development, validation, and reliability. *Eğitim Teknolojisi Kuram ve Uygulama, 7*(1), 158-179.

Kurkovsky, S. (2013). Mobile game development: improving student engagement and motivation in introductory computing courses. *Computer Science Education*, *23*(2), 138-157.

Ladislav, v & Stoffová, V. (2019). Learning Object-Oriented Programming by Creating Games. In *The International Scientific Conference eLearning and Software for Education* (Vol. 1, pp. 20-29).

Lau, W. W., & Yuen, A. H. (2009). Exploring the effects of gender and learning styles on computer programming performance: implications for programming pedagogy. *British Journal of Educational Technology, 40*(4), 696-712.

Law, K. M., Lee, V. C., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education, 55*(1), 218-228.

Leutenegger, S. & Edgington, J., 2007. A games first approach to teaching introductory programming. *ACM SIGCSE Bulletin, 39*(1), pp.115–118.

Long, J. (2007). Just For Fun: using programming games in software programming training and education. *Journal of Information Technology Education: Research, 6*(1), 279-290.

Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2014). Educational games for teaching computer programming. In C. Karagiannidis, P. Politis, & I. Karasavvidis (Eds.), *Research on e-learning and ICT in education* (pp. 87–98). New York: Springer.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004, January). Scratch: a sneak preview [education]. In *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004.* (pp. 104-109). IEEE.

Mihci, C., & Ozdener Donmez, N. (2017). Teaching GUI-Programming Concepts to Prospective K12 ICT Teachers: MIT App Inventor as an Alternative to Text-Based Languages. *International Journal of Research in Education and Science*, *3*(2), 543-559.

Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information Technologies*, *7*(1), 55-66.

Montes, H., Hijón-Neira, R., Pérez-Marìn, D., & Montes, S. (2021). Using an Online Serious Game to Teach Basic Programming Concepts and Facilitate Gameful Experiences for High School Students. *IEEE Access*, *9*, 12567-12578.

Morrison, G. R., Ross, S. M., & Kemp, J. E. (2004). *Designing effective instruction, 4th edition*, New York, NY: John Wiley & Sons Inc.

Mullins, P. M., & Conlon, M. (2008, October). Engaging students in programming fundamentals using Alice 2.0. In *Proceedings of the 9th ACM SIGITE conference on Information technology education* (pp. 81-88).

Nevalainen, S., & Sajaniemi, J. (2005). Short-Term Effects of Graphical versus Textual Visualisation of Variables on Program Perception. In *PPIG* (p. 8).

Nolan, K., Bergin, S., & Mooney, A. (2019, September). An Insight Into the Relationship Between Confidence, Self-efficacy, Anxiety and Physiological Responses in a CS1 Exam-like Scenario. In *Proceedings of the 1st UK & Ireland Computing Education Research Conference* (pp. 1-7).

Pallant, J. (2015). *SPSS survival manual: A step-by-step guide to data analysis using IBM SPSS*. Routledge.

Papadakis, S. (2018). The use of computer games in classroom environment. *International Journal of Teaching and Case Studies*, *9*(1), 1-25.

Pattis, R. E., Roberts, J., & Stehlik, M. (1995). Karel the Robot: A Gentle Introduction to The Art of Programming.

Petrovskaya, E. (2019). *Not All Fun and Games: The Design and Evaluation of a Game to Increase Intrinsic Motivation in Learning Programming*. (HCI-E MSc Final Project Report). University College, London.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004, June). Self-efficacy and mental models in learning to program. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 171-175).

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., & Lahmine, S. (2015). Learning basic programming concepts by creating games with scratch programming environment. *Procedia-Social and Behavioral Sciences, 191*, 1479-1482.

Owolabi, J., Olanipekun, P., & Iwerima, J. (2014). Mathematics ability and anxiety, computer and programming anxieties, age and gender as determinants of achievement in basic programming. *GSTF Journal on Computing (JoC)*, *3*(4), 109.

Özdener, N. (2008). A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer-programming students. *Computers & Education*, *51*(3), 1094-1102.

Seberová, A., & Malčík, M. (2014). Information System "Diagnostic" as a Tool of Action Research. *International Journal of Information and Communication Technologies in Education*, *3*(1), 57-65.

Seng, W. Y., Yatim, M. H. M., & Hoe, T. W. Learning Object-Oriented Programming Paradigm via Game-Based Learning Game–Pilot Study. *The International Journal of Multimedia & Its Applications (IJMA) Vol.10*, No.6, December 2018

Seralidou, E., & Douligeris, C. (2021). Learning programming by creating games through the use of structured activities in secondary education in Greece. *Education and Information Technologies, 26*(1), 859-898.

Soares, A., Martin, N. L., & Fonseca, F. (2015). Teaching introductory programming with game design and problem-based learning. *Issues in Information Systems, 16*(3).

Spires, H. A., Rowe, J. P., Mott, B. W., & Lester, J. C. (2011). Problem-solving and game-based learning: Effects of middle-grade students. Hypothesis testing strategies on learning outcomes. *Journal of Educational Computing Research, 44*(4), 453–472.

Song, C., Wang, H., Yang, B., & Zhang, W. (2020, September). Online and Offline Teaching Mode of C Language Programming. In *Proceedings of the 2020 The 2nd World Symposium on Software Engineering* (pp. 207-210).

Theodoraki, A., & Xinogalos, S. (2014). Studying students' attitudes on using examples of game source code for learning programming. *Informatics in Education, 13*(2), 265-277.

Thota, N., & Whitfield, R. (2010). Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education, 20*(2), 103-127.

Topalli, D., & Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers & Education, 120*, 64-74.

Vassilev, T. I. (2015). An Approach to Teaching Introductory Programming for IT Professionals Using Games. *International Journal of Human Capital and Information Technology Professionals (IJHCITP), 6*(1), 26-38.

Vahldick, A., Mendes, A. J., & Marcelino, M. J. (2014, October). A review of games designed to improve introductory computer programming competencies. In *2014 IEEE frontiers in education conference (FIE) proceedings* (pp. 1-7). IEEE.

Wong, Y. S., Hayati, M. Y. M., & Tan, W. H. (2016, September). A propriety game-based learning game as learning tool to learn object-oriented programming paradigm. In *Joint International Conference on Serious Games* (pp. 42-54). Springer, Cham.

Yağcı, M. (2016). Effect of attitudes of information technologies (IT) preservice teachers and computer programming (CP) students toward programming on their perception regarding their self-sufficiency for programming Bilişim teknolojileri (BT) öğretmen adaylarının ve bilgisayar programcılığı (BP) öğrencilerinin programlamaya karşı tutumlarının programlama öz yeterlik algılarına etkisi. *Journal of*

*Human Sciences*, *13*(1), 1418-1432.

Yan, L. (2009, April). Teaching object-oriented programming with games. In *2009 Sixth International Conference on Information Technology: New Generations* (pp. 969-974). IEEE.

Zainal Abidin, N. H., Arsad, R., Muslim, N., & Masrom, S. (2020). Computer game application for JAVA programming language learning. *Mathematical Sciences and Informatics Journal (MIJ)*, *1*(1), 77-89.

Zhu, J., Alderfer, K., Furqan, A., Nebolsky, J., Char, B., Smith, B., ... & Ontañón, S. (2019, August). Programming in-game space: how to represent parallel programming concepts in an educational game. In *Proceedings of the 14th International Conference on the Foundations of Digital Games* (pp. 1-10).

## Author Information

| **Osman Gazi Yildirim** | **Nesrin Ozdener** |
|---|---|
| https://orcid.org/0000-0003-2282-5997 | https://orcid.org/0000-0002-5549-0532 |
| National Defense University | Marmara University |
| Army NCO Vocational College | Computer Education and Instruction Technology |
| Turkey | Department |
| Contact e-mail: *ogyildirim32@gmail.com* | Turkey |