Teaching programming for future information technology specialists based on end-to-end design of computer games

Svitlana O. Leshchuk[1], Ihor A. Tverdokhlib[2,3,4], Tetiana V. Pidhorna[5], Denys I. Huska[6] and Albina A. Patyashina[1]

[1] *Ternopil Volodymyr Hnatiuk National Pedagogical University, 2 Maxyma Kryvonosa Str., Ternopil, 46027, Ukraine*
[2] *Institute of Pedagogy of the National Academy of Educational Sciences of Ukraine, 52-D Sichovyh Streltsiv st., Kyiv, 04053, Ukraine*
[3] *Mykhailo Dragomanov Ukrainian State University, 9 Pyrogova Str., Kyiv, 01601, Ukraine*
[4] *Taras Shevchenko National University of Kyiv, 60 Volodymyrska Str., Kyiv, 01033, Ukraine*
[5] *State University of Trade and Economics, 19 Kyoto Str., Kyiv, 02156, Ukraine*
[6] *Digital Cloud Technologies, 31b 23 Serpnya St., Kharkiv, 61000, Ukraine*

Abstract
The article is devoted to the formation of professional and special competencies of future information technology specialists in the field of programming. The authors of the article propose to study programming through end-to-end design. The method of end-to-end design can be used only after students of computer science specialties have developed basic skills in algorithmization and programming, and have learned one or more programming languages and developed basic knowledge of software engineering in them. The authors propose to consolidate the study of programming by creating a computer game in the process of its end-to-end design. To do this, the authors developed the content of an elective course in programming, proposed a methodology for its implementation, and conducted experimental training for students of computer science specialties. A computer game is designed in the Unity environment in C#. An important part of the study is the description of the main stages of end-to-end design of a computer game, which includes code fragments and focuses on the features of this stage. In the process of the study, a survey of programming lecturers and students of computer science was conducted to find out their opinion on the importance and feasibility of introducing end-to-end design into the process of teaching programming. According to the results of the survey of programming lecturers, it was determined that the use of end-to-end design is appropriate in the older years of the university. Most lecturers use classical methods of teaching programming in their practice (individual coding, project-based learning, flipped learning), but note that end-to-end design is an important method for consolidating programming skills, developing students' ability to work in a team, project development and management skills, and developing students' systematic, logical, and abstract thinking.

Keywords
teaching, learning, method, programming, students, computer science, IT specialists, end-to-end design, computer game, C#, Unity

1. Introduction

The defining feature of the modern information society is the use of information technologies in most areas of human activity: economics, industry, medicine, service and social sphere, education, everyday life, etc. The information society is understood as a stage of development of the post-industrial society where information is the main resource and information processes determine economic, political, industrial and social activities. The information society is also characterized by the increasing role of information and knowledge, the growing role of information technology and services, the use of new forms of communication, and the increasing role of human information culture.

STEM@ICon-MaSTEd 2025: 4th Yurii Ramskyi STE(A)M Workshop co-located with XVII International Conference on Mathematics, Science and Technology Education, May 14, 2025, Ternopil, Ukraine
leshchuk_so@fizmat.tnpu.edu.ua (S. O. Leshchuk); i.a.tverdokhlib@gmail.com (I. A. Tverdokhlib); t.pidhorna@knute.edu.ua (T. V. Pidhorna); hd.enjoy.27@gmail.com (D. I. Huska); apatyashina@gmail.com (A. A. Patyashina)
0000-0001-6600-7940 (S. O. Leshchuk); 0000-0001-6301-0159 (I. A. Tverdokhlib); 0000-0002-1414-3489 (T. V. Pidhorna); 0009-0007-8444-6067 (D. I. Huska); 0009-0002-1782-0307 (A. A. Patyashina)

In the process of development of the modern information society, education plays an important role. On the one side, it is transforming towards the active use of information technologies in the educational process, and on the other side, it is the main engine of the development of information technologies and the information society in general. At the same time, an important role is assigned to the professional training of future IT specialists. They play a key role in configuring and servicing information systems, develop, configure and support the correct work of software, influence on the level of development and integration of information technologies in all spheres of human life.

An important place in the professional activities of modern information technology specialists is occupied by the skills of independent obtaining of new knowledge, skills and abilities, self-organization and regular professional development [1]. This is caused by the continuous development of information technologies, their regular updating, which takes place every few years. Therefore, there is an emergency need to train a competitive information technology specialist who is able to navigate the changing information society. Such training of future IT specialists can be provided by studying the theoretical foundations of the functioning of digital devices and forming general and professional competencies.

In the process of informatization of society, the vision of teachers and scientists about the place of programming in the school computer science course was also changing. Initially, the programmer's approach prevailed, emphasizing such concepts as "program" and "algorithm", and computer literacy was understood as the ability to program. Further study of computer science at school was based on the user approach. However, in today's information world, programming skills are an integral part of the professional training of secondary school pupils and a mandatory component of the professional competence of future IT specialists.

The issues of professional training of future IT specialists are presented in many Ukrainian and international scientists' works. In particular, A. Striuk and S. Semerikov in their scientific researches [2, 3, 4] consider the issues of forming the competencies of future software developers and ways to achieve the goals of their training. The papers [5, 6, 7, 8] described the methods and means of effective programming training. The authors in [6] compare progressive web applications with applied and hybrid programs, study the role of progressive web applications in students' learning activities, present the development of their own web application, and select resources important for students' education in the field of information technology. Paper [7] proposes a methodology for organizing students' collaboration when working on projects using the GitHub service during programming studies.

Currently, many modern techniques, methods, trends, and tools for teaching programming have been developed and published in scientific publications. A lot of attention is paid to the formation of systemic [9, 10] and logical [11] thinking of IT specialties students, and the methodology of using the project method in teaching programming [12, 13, 14]. Olena Glazunova et al [15] consider the creation and use of a cloud-based environment for flipped learning for future IT professionals. It describes the organization of the students' project work based on flipped learning using the services and resources of the cloud-oriented educational environment.

The professional training requirements for future IT specialists are laid down in the state higher education standard. The state standard in the field of Information Technology provides for the development of such general competencies as: the ability to think abstractly, analyze and synthesize; the ability to develop and manage projects; the ability to work in a team; the ability to think logically, to draw logical conclusions; the ability to think systematically. All these competencies are formed in the process of studying programming. It is worth noting that students of computer science specialties study programming for several semesters or even years. At the same time, there is a fragmentation of the topics of the general programming course. Students can study algorithms and data structures, basic algorithmic constructions, syntax of the chosen programming language, algorithms for processing data arrays, features of creating a graphical user interface, etc. in different semesters. In other words, the process of developing a software application from start to finish is usually not covered in academic programming courses.

The methodology of end-to-end design is gaining more and more popularity every year in the process of training university students. The importance of its use lies in the fact that in the process of end-to-end design, students develop systematic and algorithmic thinking, that is, the student considers the task from the point of view of a systematic approach. Moreover, the use of end-to-end design in the study of programming contributes to the development of skills in computer science students to develop a complete software application as a holistic process that involves teamwork, communication, self-help, and self-learning.

Currently, there are many scientific works devoted to the use of the end-to-end design for modeling information systems and processes. Many of them are devoted to the end-to-end design of information system components or technical devices [16, 17], computer vision systems, neural networks, and systems based on artificial intelligence [18, 19]. David Magda from the University of Florida (United States) and his coauthors present the design of a browser-based Arduino programming tool and learning management system that offers end-to-end support for learners utilizing Chromebooks in a classroom environment [20]. Thus, there are no scientific publications about the use of the end-to-end design method in the study of programming by students of computer science. Computer games have recently occupied a significant place in people's lives: people play computer games and use game technologies in education. A large number of publications by scientists from all over the world are devoted to the use of game technologies in education. Paper [21] presents a systematic review of the use of gamification in software engineering education.

Given the popularity of games and gaming technologies in education, developing computer games can be an effective way to get students interested in learning programming and an interesting result of teaching programming. You can find some scientific works devoted to the methodology of teaching computer games in higher education institutions. Mark Frydenberg in his study [22] describes a project where first-year college students in an introductory technology concepts course use a visual game creation tool to develop original games to play on their computers and mobile devices. Ladislav Végh and Veronika Stoffová in [23] consider computer programming to be one of the most important subjects for undergraduate computer science students. They describe different approaches to teaching programming. The authors believe that creating your own games is a great motivating factor for learning programming, because most students like to play computer games. In their study, the researchers propose to study object-oriented programming by developing simple games.

Thus, there are scientific works in which the authors propose to create simple games in the process of learning programming. A lot of works are devoted to modeling information systems (or parts of them) and processes using end-to-end design. However, there are no works devoted to the methodology of teaching programming students of computer science specialties based on the end-to-end design of computer games.

The purpose of the study is to develop a teaching methodology (content, structure of the elective course, features of its implementation in the educational process) for programming future information technology specialists based on end-to-end design of computer games and to experimentally test the feasibility of its implementation in the educational process by surveying programming lecturers and university students who had completed the proposed course.

2. Selection of methods and diagnostics

In this study, we used a system of methods to achieve the goal, among which the most important were the following: analysis of scientific, technical and specialized literature on the research problem, systematization, generalization (to find out the state of development of the problem and review possible ways to form professional competencies in future information technology specialists), experimental training, observation of the educational process (to implement and adjust the methodology of end-to-end design of computer games in the process of teaching programming to students of computer science specialties), questionnaires, interviews with students and teachers, analysis of the results of the pedagogical experiment (to identify the pedagogical impact of end-to-end design of computer games on the formation of competencies of students of information technology specialties).

To achieve the goals of the study, we used a survey method based on our own questionnaire for lecturers in the Google Forms environment. The following questions were asked of university lecturers in the questionnaire we developed:

1. What is your experience of teaching at the university?
2. What specialties do you teach?
3. What disciplines do you teach?
4. How, in your opinion, is it advisable to organize programming training for students of computer science specialties?
5. Evaluate the similarity of the structure of students' activities in the process of project implementation and end-to-end software design.
6. Choose the most appropriate definition of end-to-end software design.
7. Evaluate the effectiveness of each teaching methodology in teaching programming to students in the field of knowledge 12 "Information Technology" (1 - low effectiveness, 5 - high effectiveness).
8. Evaluate the impact of end-to-end design in the process of teaching programming on the formation and development of students' competencies (1 - no impact, 5 - very significant).
9. Choose the type of software whose development is of most interest to students
10. Express your opinion on the use of end-to-end design in the process of teaching programming.

To analyze the survey results, we used methods of system analysis, digital modeling, and data visualization. The survey results were processed using Google Sheets [24].

We conducted a survey of information technology lecturers using the questions presented above in more than 10 higher education institutions of Ukraine. We organized experimental teaching of information technology students on the following basis Ternopil Volodymyr Hnatiuk National Pedagogical University (Ternopil, Ukraine) and Mykhailo Dragomanov Ukrainian State University (Kyiv, Ukraine). We conducted systematic conversations with students during the experimental training. Students were surveyed to find out their opinions on the feasibility of using the end-to-end design method in learning programming, their interest in developing computer games (or other types of software), their impressions of the process of developing a completed computer game, the need to involve specialists in related fields (game designers, artists, testers, sound designers) to improve game development, etc.

3. Theoretical background

The structure of the programming course has gained a well-established form in recent years. The content and structure of the subject "Programming" is developed in accordance with the state standard, the curriculum of the specialty, the state of development of programming languages and technologies. Usually, classes are held in the form of lectures and practical lessons, where students acquire skills in coding. Different approaches to teaching programming can be used: project activities, flipped learning, writing code in small groups using cooperation or collaboration, end-to-end design, etc.

The lectures present theoretical material related to the programming process in general, the description of syntactic constructions and various tools of a particular programming language, some issues of programming technologies, etc. Lectures are actively conducted using the method of demonstration examples and the method of open programs. When discussing theoretical material, it is important to immediately point out its possible practical applications or demonstrate them. For this purpose, the process of solving a series of problems is considered, on the basis of which attention is focused on the mechanisms of operation of individual programming language tools. The teacher's independent solution of the problem is intended to demonstrate all stages of creating a software tool in the development environment: setting a task, analyzing the input data and the result, abstraction, building a solution model, finding and determining the optimal algorithm, selecting the necessary data structures, writing the algorithm using the programming language, detecting errors and debugging the software tool, testing. Creating a simple software tool, even if the program does not have an explicit functional structure, is a rather laborious process that involves at least three components: preparing the environment, coding, and debugging the program. Therefore, analyzing the process of solving a problem in such detail in a lecture has certain disadvantages - significant time consumption and possible distraction by unimportant details.

The main type of classes for acquiring practical programming skills is practical work by students. The task for a practical programming assignment usually contains a problem statement, a list of recommended tools for solving it, and a list of some reference resources. That is, the student is essentially given a problem to solve. To do this, the student needs to analyze possible ways and means to solve the problem, choose a programming language and environment. To develop students' skills of teamwork, interaction, and communication, the task should be solved as part of a team project activity. The authors in [13] state that "...the acquisition of problem-solving and communication abilities is critical in programming education. These skills are crucial for identifying, analyzing, and solving problems effectively."

Recently, an activity-based approach to the teaching process has been used in the learning of programming. It is based on the concept of "learning by doing". The basis of this approach is related to the ideas of the theory of experiential learning, according to which knowledge is acquired through the transformation of practical experience [25]. The main methods of teaching are the method of appropriately selected tasks, the method of demonstration examples, the method of open source programs, and the project-based method. Each of these methods has both positive aspects and certain disadvantages, and requires pedagogically balanced use depending on the teaching conditions.

According to the theory of cognitive load, knowledge is divided into primary knowledge (usually not learned) and secondary knowledge, the importance of which is determined by the culture of humanity. Subject-specific biologically secondary knowledge and skills need to be clearly taught and actively learned, which are usually not acquired easily, unconsciously, and automatically [26]. Creating conditions close to future professional activities in the process of theoretical training and mastering theoretical knowledge and methods of activity in these conditions is an important stage in the process of professional training. One of the ways to achieve this may be to learn the stages of end-to-end design using the example of game development. As a rule, it is difficult for IT students to determine the requirements for software used in a particular subject area (they are not specialists in this field), but the great majority of students are active users of various computer games and they can easily be customers and put forward appropriate requirements for software.

Project-Based Learning is an activity-based teaching method where students gain knowledge and skills by working for a lengthy period to examine and react to genuine, engaging, and complex questions or challenges [27].

The project-based learning approach to programming is an approach to teaching programming in which students complete holistic projects as part of acquiring practical programming skills instead of writing separate blocks on a particular topic. The use of this method in programming contributes to the formation of deep knowledge and skills of students, deepens and shapes the personal qualities of the future IT specialist, such as the ability to work in teams, planning, initiative, and the ability to find ways to solve complex problems.

End-to-end design of software offers a comprehensive solution covering all stages of product development - from the concept to its full implementation and support. End-to-end software development implies efficient and well-coordinated work in all its phases. Team members communicate closely with each other, which allows them to stick to the chosen vector and coordinate steps. End-to-end design of software consists of the following steps: requirement analysis, planning, prototyping, UI/UX design, coding, QA testing, launch, maintenance [28].

By end-to-end design in programming education, the authors of the article mean an approach to organizing teaching programming in which students study and practice all stages of the software development process using the example of software development from start to finish.

This approach to teaching programming allows students to experience the full software development cycle in practice. In the process of end-to-end design, they can see how the choice of technologies at one stage affects other stages of software development and the final result, helps to develop the ability to analyze and solve complex problems, and the skills to work independently on real projects in their future professional activities.

This approach has been used in writing well-known bestselling programming books. For example, in "Assembly language for the PC" [29], the authors create a software project throughout the book - the shell of the Norton Commander operating system, which was very popular in the 90s of the last century. Herbert Schildt in his book "Java: A Beginner's Guide" [30] creates a software project throughout the book - the Java help system.

Thus, in some aspects, the end-to-end design method resembles the project-based method, but there are significant differences in the construction of the programming course and the organization of the educational process. The project-based method involves creating a software product to accomplish a specific task, in which students apply their knowledge in practice. This can be a separate stage of learning where students apply knowledge to solve a specific problem. The end-to-end design method covers the entire life cycle of software product development. It allows students to gain skills in writing code, planning, architecture selection, testing, maintenance, and implementation of software. That is, the process of creating a software product is considered from the point of view of a systematic approach in which future IT professionals consider the software development process as a whole.

4. Results and discussion

4.1. Description of a comprehensive course for learning programming

In our study, we proposed the idea of building a programming course for future IT specialists based on end-to-end design. Of course, this cannot happen at the beginning of programming studies at university. At the beginning, students need to learn the basics of algorithmization and programming, basic code writing skills, and one or more programming languages. The learning of programming in the first years of university is usually accompanied by writing separate parts of the code to study the basic algorithmic structures, syntax, and semantics of the programming language, and by performing small-scale learning projects to master specific skills or technologies (developing a user interface, developing an authorization system, creating and connecting a database, etc. These projects usually do not involve designing a software product from the initial idea to implementation and support.

In our proposed course "Computer Game Development Technologies" or "Computer Game Programming Technologies" (the course had different names in some universities), the process of creating a computer game using the end-to-end design method was studied. This subject was taught in the 3-4th year of study and was the last subject of the programming cycle. The course, developed and implemented in the educational practice of training future IT professionals, is studied as an elective discipline. However, in our opinion, such a course would also be useful in the normative component of the educational program for training future IT specialists. The course content offers an approximate number of hours and weeks to study the subject. We believe it is appropriate to leave it to the discretion of graduating departments to integrate this course into the professional training program and the number of hours to study it. Its study allows students to complete the study of programming through the end-to-end design of a computer game, generalizing and systematizing students' knowledge of software design and development.

To create a computer game in the proposed course, we used the Unity environment, a cross-platform engine for developing computer games, virtual and augmented reality [31]. Unity was chosen not only because it is one of the most popular engines for creating games, but also because Unity uses the C# language for programming, which students have already begun to learn in previous courses. Thus, integrating Unity learning into the C# learning process can be a great way to reinforce practical programming skills and empower students.

It was no accident that we chose the topic of computer games, as the video game industry is very popular among young people. As you know, the success of a game depends on many factors, such as the quality of gameplay, graphics, story, marketing strategy, etc. With the development of technology, game engines, development tools, platforms, and development methods are constantly changing. With the advent of virtual reality, augmented reality, and other innovative technologies, new opportunities and challenges for game developers are becoming relevant. All this prompted us to develop and implement a methodology for training IT specialists in the gaming industry.

The proposed educational course is designed for 17 academic weeks. Below is a list of tentative course topics and their distribution over the course of the semester:

Topic 1: Project planning (Week 1 - 2). Features of end-to-end software design. Stages of work on the project, task setting, distribution of responsibilities, start of game design. *The goal* of this topic is to familiarize students with the main stages of end-to-end design of a computer game, update C# programming skills, form teams, assign responsibilities, and set tasks for the project.

The result of studying these topics is the formation of students' ability to set a task, draw up technical documentation for the project, distribute responsibilities in their team, determine the genre of the game and its concept (including the visual implementation of ideas).

Topic 2. Unity C# programming basics (weeks 3 - 5). Creating simple scripts for managing objects, interacting with components (e.g., character movement, animation control), working with events.

Goals: To introduce students to the features of the Unity interface and its main components, to demonstrate the possibilities of using C# to write scripts in Unity, integrating the basic knowledge of C# from the previous semesters.

Results: Students are able to write simple C# scripts for Unity and understand how the code interacts with Unity components, finalize the art concept (formation of visual ideas and style).

Topic 3. Designing and creating simple game mechanics (weeks 6 - 9). Using C# to develop game mechanics, creating simple animations and working with UI.

Goals: To teach students how to create simple game mechanics (e.g., movement of the main character, physics, basic interaction with objects), develop a control system, and test the developed mechanics.

Result: Students use C# to control the gameplay and interact with the user.

Topic 4. Improving a computer game (weeks 10-13). Work on graphics, sound, and game levels.

Goals: To realize graphic resources, develop a soundtrack and game levels.

Results: Students create, test, and optimize project graphics; work with the game's sound effects; programmatically implement elements of game complexity, making it more interesting.

Topic 5. Completion of the project (weeks 14-17). Testing, debugging, release of the game.

Goals: To identify and eliminate the game's shortcomings, prepare the development for release.

Results: Students complete the project, gain teamwork experience, and present the finished game.

The developed course program is designed for one semester, which allows you to effectively combine C# knowledge with the basics of Unity, while developing students' practical skills and implementing an end-to-end design approach to computer games.

In general, the implementation of the end-to-end design method when building a programming course involves several stages of implementation. The preparatory stage is carried out by the lecturer and involves a number of activities:

- based on current state standards and curricula, the relevant content of the educational material is formed;
- design and describe educational software tools, for the creation of which all the tools of the programming language and the stages of creating a software tool that are reflected in the content of the educational material should be used;
- the number of such educational software tools should be sufficient for each student to work independently;
- each educational software tool should be meaningful (not formalized), practically relevant and interesting;
- in accordance with the content of the course material, a series of lectures is prepared, which involves grouping theoretical material and demonstrating its use in the development of the educational software tool by the teacher;
- laboratory works are prepared, the content of which corresponds to the cycle of lectures; each work should contain sufficiently detailed educational and methodological instructions for imple-

mentation, on the basis of which the student will develop a working component of the future educational software tool.

Each lecture class or group of such classes should consist of two main parts:

- the lecturer explains the planned theoretical material related to the programming language tools, the programming process itself, programming technology and methods, or outlines the boundaries of the theoretical material and indicates the relevant available information sources;
- the lecturer demonstrates the application of the described theoretical material on the example of building a separate component of the educational software tool that is created throughout the course.

During such lectures, students actively learn new language tools and become participants in the process of programming and creating a real software tool. Thus, from lecture to lecture, the software tool being created is constantly being improved and acquires new, necessary functionality. And students observe with their own eyes all the stages of the real process of creating a software tool and the expediency of using certain language tools, programming methods and selected classical algorithms.

During practical work, students apply the theoretical and reference material presented in lectures to build their own project in accordance with the developed teaching guidelines. The result of each such work should be the assimilation and processing of theoretical material, which can be tested with the help of prepared test tasks. A working project should be created at this stage of development due to the added functionality based on the information received about new language tools. The teacher may give advice and consultations on planning and implementing their own projects when students perform specific practical work.

At the end of the course, the lecturer (in lecture classes) and students (in practical/laboratory classes) complete the creation of educational projects - working software tools. After that, it is necessary to hold a final meeting to demonstrate their own developed programs and a general evaluation of the work of each developer according to the criteria specified by the teacher, in which all project developers, i.e. students of this training course, will participate.

4.2. Description of the stages of game creation

Let's consider the main stages of game development, starting with gameplay design and ending with its testing, using the example of a game created by the lecturer during the lecture classes in the process of experimental programming teaching. We will describe the peculiarities of each stage and identify the key aspects that need to be taken into account.

4.2.1. Designing a game

At the beginning, the game is designed. This is one of the most important stages of game development, as it determines the further success of the game. At this stage, the gameplay, the concept of the game, its main mechanics and functionality are determined, and the basic rules are established. When designing a game, you need to consider the following issues: game structure, mechanics, levels, main hero, enemies, location, and other components that can affect the player's experience. In Table 1, you can find the main components of preparing to create a computer game, including: idea, target audience, game genre selection, general concept development, competitors' analysis, and technical capabilities assessment.

To complete the tasks of the preparatory stage, students are organized into teams to design a game. Next, they define the basic rules of the game, such as the rules of interaction between characters and the environment, combat mechanics, and others. It is important that the rules of the game are clear and logical, and not too complicated for the players. The next step is to create a game prototype that will help test the gameplay and identify possible problems.

At the design stage of the game "TopDownShooter", it was decided to create a 2D shooter with a top view, where the main character fights with enemies in separate rooms. This solution allows you

Table 1
Structure of the preparatory stage of computer game development.

Name of the component	Description
Game idea formulation	The starting point for creating any game. It can come from the imagination of developers, inspiration from other sources, feedback from players, or be based on trends and market needs. It is important that the idea is original, unique, and interesting, able to attract attention and interest of the audience
Identifying the target audience	Understanding the target audience is a key factor in the successful development of a game. It is necessary to find out for whom the game is being created, what interests and preferences this audience has, and what game elements will be most attractive to this segment of users
Choosing a game genre	The game genre determines the basic structure and mechanics of the gameplay. It can be action, strategy, RPG, puzzle, simulator, etc.
Development of a general concept	At this stage, the general concept of the game is formed. The plot, characters, atmosphere, and other aspects of the game are determined. It is worth analyzing the main characteristics, mechanics, gameplay, and unique features of the game. It is important that the game concept is coherent and logical, as well as attracts the attention of the target audience
Competitor analysis	Research and analysis of competitive games helps to determine the advantages and competitive features of the future game
Assessment of technical capabilities	Development of technical documentation

to create a rich and dynamic atmosphere where the player feels the adrenaline of each fight, using an automatic pistol, shotgun and rocket launcher to clear levels, etc.

4.2.2. Creating an art concept

To create an art concept for a game, students need to focus on forming a general visual idea and style, which will later become the basis for creating graphic resources. The art concept is the stage where the atmosphere and mood that you plan to convey to the players are visualized. It is important to make the game world attractive, recognizable, and understandable, setting the tone that will keep players engaged and excited.

Next, we suggest that you critically review examples of visual design of similar games, character concepts, and game objects. This task encourages you to explore styles that could make the game unique, as well as maintain simplicity and clarity, which is important for a mobile game project. The key aspects of this area are: selecting a color palette, determining the style of the enemies' appearance, designing rooms and effects that would create the appropriate atmosphere.

In "TopDownShooter", the enemies are colored dark green to make them look like zombies - menacing and gloomy, but clearly separated from the protagonist and his allies (figure 1). The hero of the game is a dynamic character, able to adapt to changes on the battlefield, but at the same time recognizable and distinct from the enemies.

4.2.3. Developing game mechanics

At this stage, we move directly to the practice of programming. The development of mechanics can be divided into several stages, namely:

1. Determining the genre of the game (strategy, shooter, platformer, etc.) and its main mechanics (combat, puzzles, etc.).
2. Development of the main mechanics (for a shooter, it can be a battle with enemies, and for a platformer, it can be the movement of the hero through the levels).

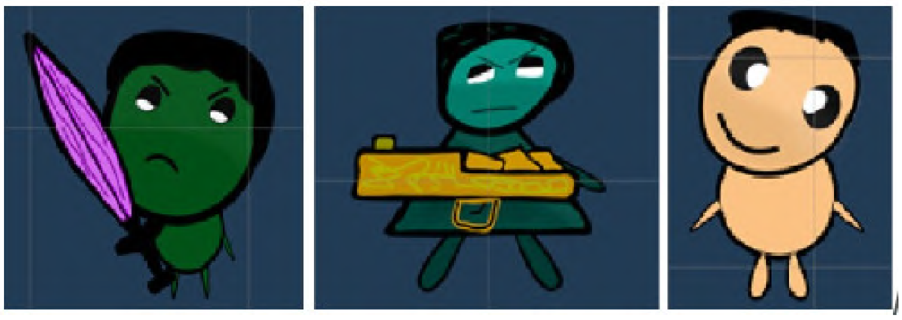
Figure 1: The look of the characters in the game "TopDownShooter".

3. Development of secondary mechanics (availability of various types of weapons; game modes, thinking through obstacles and traps on the levels).
4. Development of the control system (control of the main character, interaction with various objects on the levels, etc.).
5. Testing game mechanics (removing bugs, determining the correct interaction).

It is important for students to check whether the game mechanics work correctly, to assess whether they are interesting for players and not complicated to the point that they may reduce interest in the game. In addition, the development of game mechanics should also be done with the specifics of the platform for which it will be created. For example, game mechanics for mobile devices should be more intuitive and easy to control than those for a console game. In addition to the main stages of developing game mechanics, it is also important to ensure that they are balanced and interact with other elements of the game, such as story, graphics, and sound. All elements of the game should complement each other and create a coherent gameplay that will be interesting and engaging for players.

To implement the mechanics of the game, "TopDownShooter" focuses on the development of dynamic gameplay for mobile devices using interactive controls and various stages of difficulty. Let us consider the main mechanics of the game under development.

The movement of the protagonist is controlled by the joystick on the screen, which provides easy control on a mobile device. The player can quickly change direction, leaving one hand free for other actions.

Realization of the main character's movement:

```
private void MovePlayer(Vector2 direction)
 {
_rigidBody2D.velocity = direction * Speed;
var isFlip = direction.x > 0.0f;
_headSprite.flipX = isFlip;
_bodySprite.flipX = isFlip;
_animator.SetFloat(AnimatorSpeedId, direction.magnitude);
 }
```

Shooting is realized through a second joystick that allows you to determine the direction of fire. The weapon automatically follows the direction of this joystick, providing accurate aiming and quick response to changing enemy positions.

The procedure for obtaining weapon data:

```
public void MakeShoot()
 {
if (FireRateSeconds <= 0f && MagazineAmount > 0)
{
    FireRateSeconds = GunSettings.fireRate;
        FireRateSecondsChanged?.Invoke(1f);
        Shoot?.Invoke(GunSettings.shootSound);
    MagazineAmount--;
        CountBulletsChanged?.Invoke(MagazineAmount);
    Shooting();
 }
 }
```

Interaction with rooms. When the player enters one of the rooms, the door automatically closes, preventing escape and forcing the player to focus on clearing the room of enemies. The door opens only after the room is completely cleared, which adds an element of strategy and a sense of accomplishment.

Spawn enemies in an empty, newly entered room:

```
private void FirstSpawnEnemies()
{
    if (CurrentRoom is { IsEnemiesSpawned: false, IsCleared: false, })
    {
    CurrentRoom.EnemyWavesCount =
    GlobalProvidersLocator.GameProvider.EnemyWavesCount;
    CurrentRoom.OnEnemyWavesCountChanged();
    CurrentRoom.SpawnEnemies();
 }
}
```

Waves of enemies in each room. Each room is filled with several waves of enemies, where the number of waves and enemies depends on the current level. This adds complexity and challenges to each new level.

Check the number of waves of enemies and spawn new ones:

```
if (_enemiesCountPerWave <= 0)
  {
      IsEnemiesSpawned = false;
      EnemyWavesCount--;
      var enemyWavesCount = EnemyWavesCount;
      if (enemyWavesCount < 0)
      {
          enemyWavesCount = 0;
      }
          OnEnemyWavesCountChanged(enemyWavesCount);
      if (EnemyWavesCount >= 0)
      {
          SpawnEnemies();
      }
      else if (!IsCleared)
      {
          OpenDoors();
      }
  }
```

Level system. The game has many levels with no maximum limit. Each new level increases the number of enemies and waves that the player has to fight. The number of rooms also increases, but does not exceed 20 to ensure the optimal duration of game sessions.

Leveling up (coin accrual):

```
public void LevelUp()
 {
    var coinsPerLevelCount = Level * Random.Range(Constants.MinCoinsMultiplier,
           Constants.MaxCoinsMultiplier);
     var defaultCoinsPerLevelCount = Random.Range(Constants.MinCoinsPerLevel,
          Constants.MaxCoinsPerLevel);
     PlayerData.Coins += (uint)coinsPerLevelCount +
           (uint)defaultCoinsPerLevelCount;
     Level++;
     UpdateInfo();
 }
```

Shop for improvements. The game has a shop where you can improve the damage of weapons and buy ammunition for each type of weapon. This allows the player to better prepare for challenges and provides additional opportunities for progress.

Weapon damage upgrades:

```
protected override void Upgrade(GunType type)
{
    if (PlayerData.Coins >= Price)
    {
    var gunDataModel = GlobalProvidersLocator.WeaponProvider.
        GetGunDataModel(type);
    if (gunDataModel is not null)
        {
           gunDataModel.Damage += CountUpgrade;
         SettingsProvider.SaveSettingsByKey(type.ToString(), gunDataModel);
           OnUpgraded(UpgradeType.Damage, gunDataModel.Damage);
           PlayerData.Coins -= Price;
     }
 }
 else
 {
     Debug.Log("Not enough money");
 }
 }
```

Sound settings. In the game settings, you can turn off music and sound effects, which adds to the comfort of players.

Turn the sound on/off:

```
public void OnSoundClick()
 {
    MainMenuLocator.MainMenu.PlayAudioSource();
     SettingsData.IsSoundOn = !SettingsData.IsSoundOn;
     disableSoundImage.enabled = !SettingsData.IsSoundOn;
 }
```

Game currency (coins). The player receives coins for neutralizing enemies and completing levels. This currency allows you to buy ammunition and upgrades, which motivates you to complete more levels and take part in battles.

Bosses of every fifth level. After every fifth level, a special room with a boss appears. The character is transported to this room where he has to fight the boss, adding a special challenge and variety to the game.

Boss spawn:

```
if (!isBossSpawned && bossFactory is not null
    && message.NewRoom?.RoomType == RoomType.Boss)
        {
            isBossSpawned = true;
           var spawnPoint = PositionHelper.GetPointWithMaxDistance(Vector2.zero,
                        message.NewRoom.EnemySpawnPoints);
             Boss = bossFactory.CreateBoss(0, spawnPoint);
            GlobalProvidersLocator.Messenger.Send(new BossSpawnedMessage());
        }
```

Shield and health boosts. The game has boosts that provide a temporary shield or restore the health of the main character. The shield protects against damage for 1 minute, reducing its time with each hit. The health boost adds 30 HP. Shields can be picked up several times, which extends their effect.

Random spawn of boosts. In each room and after the completion of waves of enemies, boosts will randomly spawn. This provides additional resources to the player at unexpected moments of the game.

Boost spawn:

```
public void SpawnBoosts(List<Transform> spawnPoints)
 {
 if (spawnPoints != null)
 {
        var chanceToSpawnBoost = (int)(Random.Range(0f, 1f) * 100);
     var boosts = Boosts.Where(boost =>
             chanceToSpawnBoost > boost.Value.minPercentageForSpawn);
     foreach (var boost in boosts)
     {
         var indexSpawnPoint = Random.Range(0, spawnPoints.Count);
             GamePlayLocator.BoostFactory?.Create(boost.Value.type,
                spawnPoints[indexSpawnPoint].position);
     }
 }
 }
```

User Interface (UI). For the player's convenience, the UI includes a display of the current wave level, the number of enemies in the room, the number of rooms in the level, and the number of rooms already cleared. This helps the player to track their progress and plan the game strategy.

The use of the described mechanics allows us to create an exciting and dynamic gameplay adapted to mobile devices and ready to provide players with unforgettable adventures on each new level.

4.2.4. Creating graphic resources

At this stage, graphic assets such as characters, tildes, and other game objects are created. This may include using off-the-shelf elements such as sprites and textures, or creating new graphic assets from scratch. Work on the project involves:

1. Identify the necessary graphic resources (list them).
2. Creating layouts of graphic resources (working with sketches (ballpoint pencils) that help visualize the final product. It is advisable to use such tools as Adobe Photoshop, Sketch, Figma).
3. Development of final versions of graphic resources (taking into account file size, resolution, format and other parameters).
4. Testing of graphic resources (on different devices, at different resolutions and formats).

5. Optimize graphic assets (reduce file size, use special formats, or reduce the number of details in graphic elements).
6. Documentation of graphic resources (creating descriptions and instructions that help students to use graphic resources correctly when completing similar projects: what graphic elements were used, their size, format, and characteristics; what steps to take to add graphic elements to the game; and recommendations for editing and using graphic resources).

At the stage of creating graphic resources, you work on the design of the main elements of the game, such as the main character, enemies, weapons, levels, boosts, and interface. Characters are created in Adobe Photoshop and/or Adobe After Effects. An interesting stage is the experimentation with different styles and techniques to achieve the desired visual result and match the theme and style of the chosen genre.

In "TopDownShooter", the work on creating the design of game elements was long: different variants of graphics were tested, new details were added, colors and shapes were changed until the desired result was achieved. This process included reviewing similar characters and getting feedback from testers who helped to objectively evaluate the design, offering their own ideas that led to the improvement of the images (figure 2). The "enemies" were supposed to create an atmosphere of threat and increase the sense of danger. Two types of enemies were developed: zombies with swords for close combat and zombies with guns for ranged attacks. This gave the game a more tactical approach, where the player is forced to adapt to different types of enemies. The game also features a boss that was created intuitively by experimenting with its shapes, sizes, and colors until the image corresponded to the idea of a powerful enemy. Bonuses in the form of shield and health have become an important part of the graphic design of game resources. They are created in the form of jars that the player can pick up to get the appropriate boost. These boosts not only strengthen the player, but also add elements of strategy, as the player must plan his actions based on the availability of these resources. The shield provides temporary protection by absorbing damage, and the health boost restores part of the character's HP, allowing him to resist enemies for a longer time.

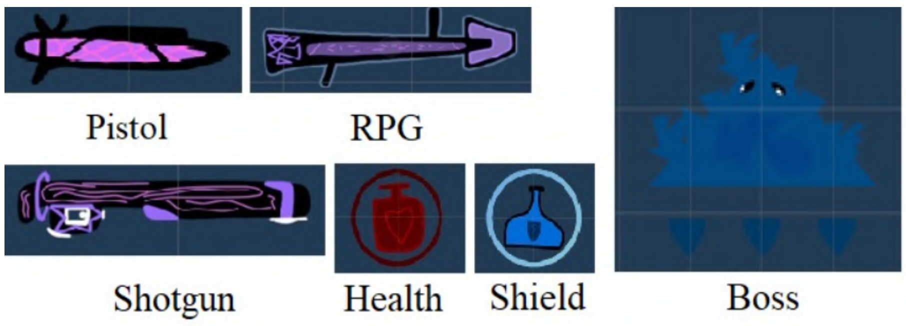

Figure 2: Design of elements of the game "TopDownShooter".

During the development of our game, special attention was paid to its interface, which should be harmoniously combined with its style and not distract the player from the main events on the screen. All interface elements - from the number of enemies to the health level - were designed so that the information is presented conveniently and is always available to the player. The creation of graphic resources allows you to create a holistic gaming environment that immerses the player in the atmosphere and enhances interaction with each element of the game.

4.2.5. Developing levels

At this stage, individual levels of the game are created, which include objects, obstacles, and other elements with which the player can interact. Level development may vary depending on the chosen development methodology, but in general, at this stage, graphic resources are created, objects and obstacles are added, and the level and protagonist logic is configured. The main goal is to create game levels that will be interesting for players and meet the general concept of the game.

The main steps of this stage:

1. Determining the general concept of the game (whether the level meets its requirements and style: obstacles and traps (for a platformer), terrain to explore, and lots of NPCs (for an RPG)).
2. Determine the type of level (timed levels, survival levels, resource gathering levels, etc.).
3. Create a level map. It should be clear and logical, with clearly marked zones and paths, as well as the locations of enemies and objects to be collected.
4. Adding elements (considering the balance (various obstacles, enemies, traps, objects to collect, and other elements that will make the game more interesting and evoke different emotions in players) between the complexity and interest of the game, so as not to frustrate players with too difficult levels or bore them with too simple ones).
5. Testing (for bugs and compliance with the overall game concept) and debugging (changing the difficulty, placement of objects and obstacles, and other adjustments).
6. Repeating for the next levels (adherence to the general concept of the game and ensuring the variety of levels).
7. Final testing (removing bugs and ensuring smoothness of the levels).

Generally speaking, level design is an important part of computer game development that requires attention to detail, student creativity, and knowledge of player psychology. Successful completion of this stage should result in a fun and engaging game with an appropriate level of difficulty. At this stage, a system of game rooms is created (figure 3), which serve as a battlefield for the protagonist, where he destroys waves of enemies, passing from one room to another. These rooms have different door configurations that not only visually distinguish them, but also change the paths and possible directions for the player, adding variability to the gameplay.

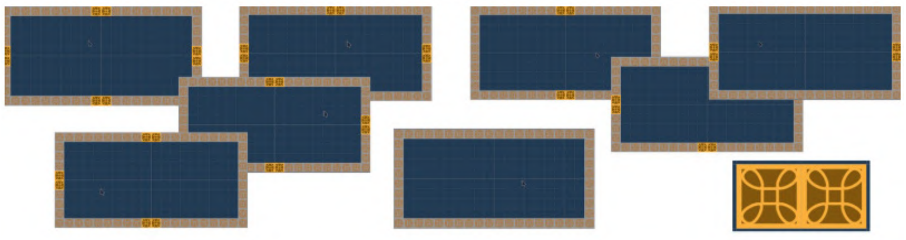

Figure 3: View of the “TopDownShooter” rooms.

The game “TopDownShooter” has the following types of rooms:

- rooms with four doors - fully open on all four sides, provide the most free movement and create dynamic conditions for combat;
- rooms with three doors - have a restriction (no door) on one side, which varies: left, right, top or bottom. This adds to the strategy as the player is forced to adapt to new space conditions;
- rooms with two doors - there are only two directions to exit (e.g., up and left, down and right, etc.), which creates even more restrictions and opportunities for tactical maneuver;
- rooms with a single door - these are the narrowest rooms to choose from, as they provide only one way in or out. They add a sense of confinement and make for intense battles;
- a room without a door for bosses is a special room that appears at the end of every fifth level. The player enters this room to fight the boss, and the only way out of the room is after the victory.

Levels are generated based on the player’s progress, and the number of rooms, waves, and enemies changes with each new level. At the initial levels, the player faces fewer rooms, waves and enemies in each wave, which opens up the opportunity to gradually adapt to the game mechanics. However, as you progress, the number of rooms increases, approaching a maximum of 20 rooms. The number of waves of enemies in each room and the number of enemies in each wave also increase depending on the level, which creates a greater challenge and encourages the player to improve their skills, as well as use boosts and the magazine to upgrade weapons and replenish ammo.

4.2.6. Developing a soundtrack

At this stage, sound effects and music for the game are created. This usually includes creating sounds for various actions of the protagonist, sounds of obstacles and other objects, as well as musical compositions that can be used in the game. A well-chosen soundtrack can emphasize the emotional component of the game and create a certain atmosphere. In order to create a high-quality soundtrack, we suggest that students follow these steps:

1. Identifying needs (game atmosphere: if it's an action game, you can use loud and dynamic sounds to create a more dynamic atmosphere and suspense, and in the case of a game with a serious storyline, you can use more subdued and emotional sounds, in the case of a puzzle game, you can use sound effects that emphasize the unpredictability and complexity of the game).
2. Choosing music and sound effects (selecting freely available sound resources, creating your own, taking into account the volume and tempo, and considering interactivity).
3. Integration of the soundtrack into the game (sound balance, technical limitations, testing, optimization, documentation, editing, copyright, testing, optimization, implementation).

At this stage of development of TopDownShooter, it was important to develop a system of sound effects (explosion sound, coin collection sound, impact sound, etc.). To do this, students were offered environments such as Audacity or Adobe Audition. You can add sound files and effects by programming or using game development tools that allow you to add sound to game objects, actions, or events.

When testing the soundtrack, you need to make sure that the sounds are reproduced with high quality and correspond to the game events. In addition, it is important to take into account technical limitations (for example, some devices may have restrictions on the number of sound tracks or their frequency range).

When optimizing, we suggest that students make sure that the size of the sound files is not too large and does not take up too much space on the hard disk. It is also important to pay attention to the number of game objects that play sound and reduce them if they are too large. This will help to conserve computer resources and ensure that the game runs quickly.

There are two different melodies for the background of TopDownShooter. The first one for the main menu is calm but intriguing; it creates a sense of anticipation and sets the player up for future adventures; the second one for the playing field is dynamic, emphasizes the tension of the battle and stimulates the player to act quickly. To find both tracks, various resources were explored in an effort to find compositions that would match the overall atmosphere of the TopDownShooter game. After testing several options, we selected the ones that were best combined with the graphics and gameplay dynamics.

Sound effects for shooting also play an important role in the game. Different sounds were selected for each type of weapon to make each shot sound unique and match the character of the weapon.

4.2.7. Testing, debugging, releasing the final version of the game

During the testing phase, students are asked to perform a series of tests to verify the game's functionality and identify errors and bugs. It is advisable to use a variety of tools, such as automatic testing frameworks, manual testing, programs for collecting and analyzing performance data, etc. It is also necessary to allocate enough time and resources to support the game to ensure its quality and correct operation in the conditions of changing operating systems and devices. In addition, it is important to ensure that bugs and crashes in the game can be tracked in order to respond quickly and release appropriate patches and updates.

Before releasing a game, it is necessary to check that it works on different computer and device configurations and that it does not use an excessive amount of system resources. For the successful release of the game, it is also important to take into account the platforms on which the game will be launched (PC, game consoles, mobile devices), the need to support different languages and localize the game for different regions, protection against hacker attacks and fraud.

Testing and debugging of TopDownShooter allowed us to thoroughly check the game's performance, improve the gameplay and eliminate the interface defects. To hear students' impressions of the game's design, peculiarities of the game design process, potential problems, and exchange of impressions, we conducted a survey of students.

4.3. Methodology of organization and results of the experimental research

The authors' research is aimed at developing a programming course for students of computer science. The peculiarity of this course is that it is taught using the technology of end-to-end software design. In our case, a computer game.

To develop the content of the course, we analyzed the state standard of higher education, curriculum, and study programs for training future computer science specialists. The analysis of scientific and educational literature made it possible to identify the psychological and pedagogical features of organizing the study of programming by means of end-to-end design.

At the development stage of the course, we conducted a survey of university lecturers using a questionnaire we developed. The purpose of the survey was to determine the opinion of programming lecturers on the feasibility of using certain methods and tools for teaching programming, their understanding of the end-to-end design method and the feasibility of using it in the educational process. The survey involved 30 lecturers from more than 10 higher education establishments in Ukraine.

The high reliability of the survey results can be evidenced by the fact that 73.3% of respondents have more than 10 years of experience in higher education institutions, namely: 50% of respondents have more than 20 years of experience, 23.3% of respondents have more than 10 years of experience. This indicates their great pedagogical experience, good awareness of the peculiarities of organizing the educational process and a deep understanding of the methods of teaching programming.

As a rule, lecturers of professional subjects in the information technology field teach their subjects in different specialties. Our survey involved lecturers who teach in the vast majority of the following specialties: "Computer Science, Secondary Education (Computer Science), and Professional Education (Digital Technologies). At the same time, the vast majority of the surveyed teachers (93.3%) teach the subject "Programming" (figure 4).

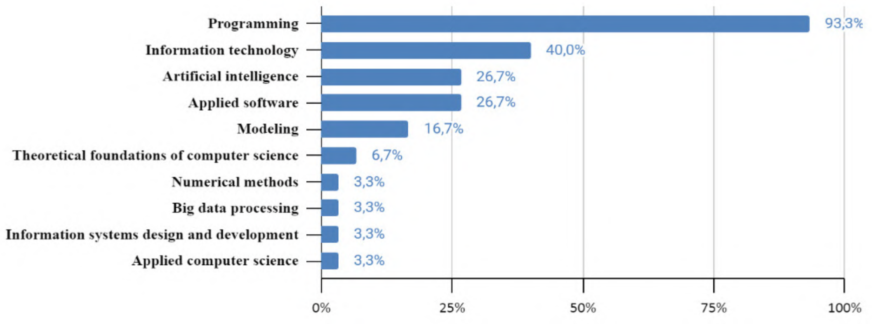

Figure 4: Bar chart visualizing the answers of respondents to the questions "What subjects do you teach?".

The lecturers were asked to answer the question "How, in your opinion, is it advisable to organize programming training for students of computer science specialties?" (figure 5). This question could be answered with several options (several methods). The respondents preferred the option "Performing individual tasks within the study of a particular subject" (90% of answers). This can probably be explained by two factors. Firstly, the question did not specify the stage of programming study for which we choose the learning method. Indeed, at the beginning of learning programming, it is advisable to write code for small tasks to consolidate knowledge and skills. Another possible explanation for this choice may be the “obsolescence” of the methodological system of teaching programming by lecturers who are old and stick to the old methods of teaching programming. It is well known that a person who has been working with one methodology for a long time has a hard time switching to new, progressive teaching methods. The analysis of the rest of the lecturers’ answers to this question shows that they also prefer to use the project-based method as part of their educational practice (73.3%) and as part of the study of a particular subject (56.7%).

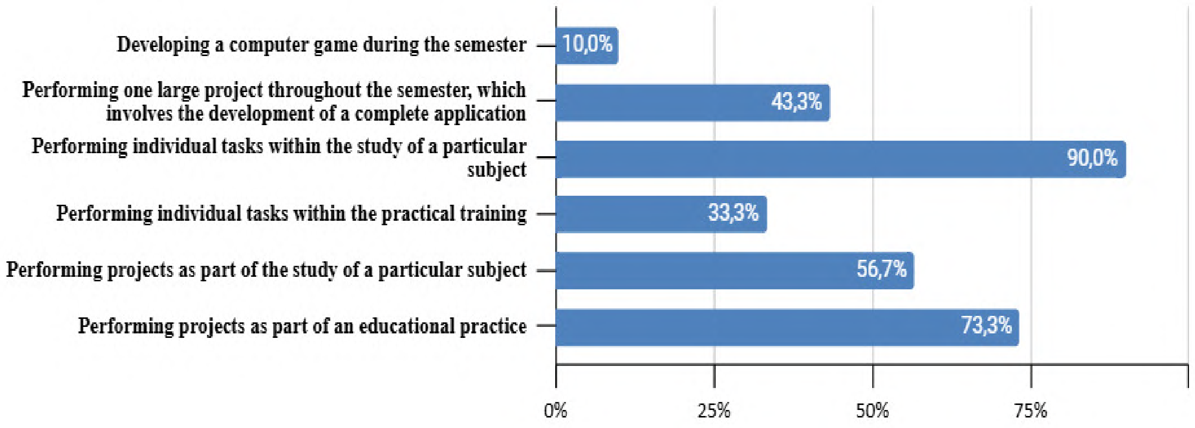

Figure 5: Bar chart visualizing the answers of respondents to the questions “How, in your opinion, is it advisable to organize programming training for students of computer science specialties?”.

The analysis of the survey results showed that only 10% of lecturers said that it is advisable to organize programming training as a computer game development during the semester, but answering another question, 33.3% of lecturers said that the development of computer games is the most interesting for students (figure 6). Moreover, another 33.3% believe that the development of a web application is the most interesting for students, and 30% - application software.

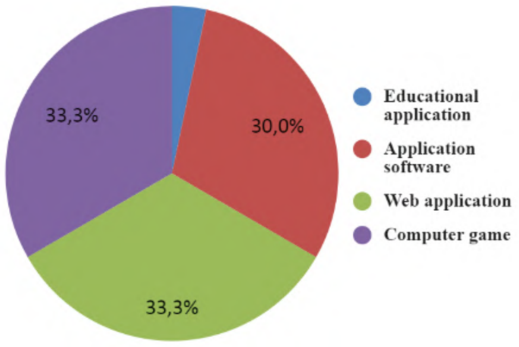

Figure 6: Circular diagram visualizing the answers of respondents to the questions “Choose the type of software whose development is of most interest to students”.

In the theoretical basis of the study, we have demonstrated some similarities between the project-based method and the end-to-end design method in the study of programming. We asked computer science lecturers to evaluate the similarities between the project-based method and the end-to-end design method. The results of their answers can be seen in figure 7. Indeed, these two learning technologies have some similarities and many distinctive features.

Lecturers were asked to evaluate the effectiveness of the proposed methods of teaching programming (individual completion of laboratory tasks, individual work on a project, teamwork of students on a project in the course of studying the discipline, end-to-end software design during the semester, flipped

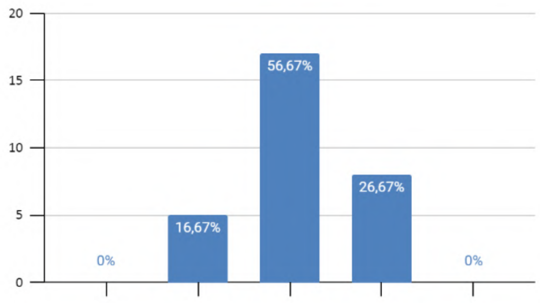

Figure 7: Bar chart visualizing the answers of respondents to the questions "Evaluate the similarity of the structure of students' activities in the process of project implementation and end-to-end software design" (1 - Completely different technologies for teaching programming, 5 - Almost identical programming teaching technologies)".

learning) on a 5-point scale: 1 - low efficiency, 5 - high efficiency. We calculated the average score for each methodology (figure 8).

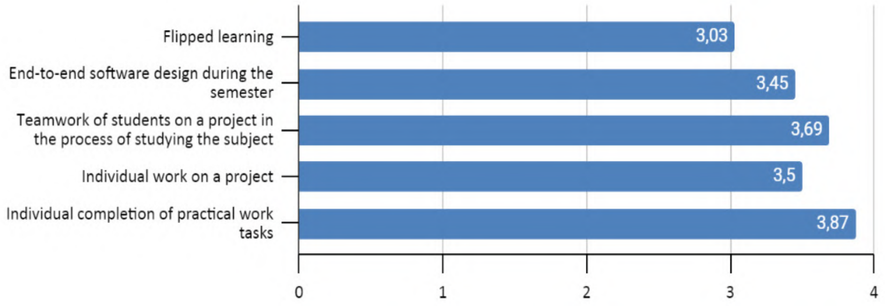

Figure 8: Bar chart visualizing the answers of respondents to the questions "Evaluate the effectiveness of each teaching methodology in teaching programming to students of computer science specialties" (1 - low effectiveness, 5 - high effectiveness).

The chart above shows that flipped learning technology has the least impact on the effectiveness of programming education. According to the respondents, the middle ground is occupied by end-to-end design of a software tool during the semester and individual project implementation. The most successful methods for use in the study of programming by students of computer science specialties are individual completion of practical work tasks and teamwork of students on a project in the process of studying the subject. It is surprising, in our opinion, that lecturers prefer individual practical work. However, this methodological approach may be appropriate at the beginning of learning programming. The effectiveness of using team-based projects has been proven in many scientific and methodological publications. Therefore, the choice of lecturers in favor of teamwork on the project is justified, as this type of activity is an effective means of forming and developing students' general and professional competencies.

In the questionnaire, we offered several definitions of end-to-end design. The most successful definition of end-to-end software design was the following: *an integrated approach to software development that covers all stages of the software life cycle from the initial idea to deployment, maintenance, and improvement.* This is exactly what we were able to implement in the developed course on learning C# in the process of creating a computer game in the Unity environment and demonstrated on the example of the development of "TopDownShooter". Students already had a basic level of knowledge and skills in programming in other languages and using editors to create media data.

Confirmation that teaching programming based on end-to-end design is an interesting and promising methodology can be found in some of the respondents' opinions, which generally confirm the idea of our study, namely: "...it is appropriate in the senior years, but not before the third year. First-year students are not ready in terms of knowledge...”, ”A good approach for older students. First (especially) and second year students will have a lot of problems (they do not have enough professional knowledge and experience in general program development)”, ”...it is advisable to use end-to-end design after studying programming technologies...”, ”A powerful tool for developing teamwork skills”, ”Given quality support from the lecturer and student interest, it may well be an effective teaching method”.

The analysis of the lecturers’ answers to the question “Evaluate the impact of end-to-end design in the process of teaching programming on the formation and development of students’ competencies (1 - no impact, 5 - very significant)” shows that in their opinion, end-to-end design has the greatest impact on the development of students’ systematic thinking, teamwork, and project development and management skills (figure 9). That is, lecturers do not consider end-to-end design as the main teaching method, but at the same time, they highly appreciate its impact on the formation of students’ competencies. This can be easily explained by the possibility of using end-to-end design in the study of programming only in the older years of study, after students have mastered basic programming knowledge and skills.

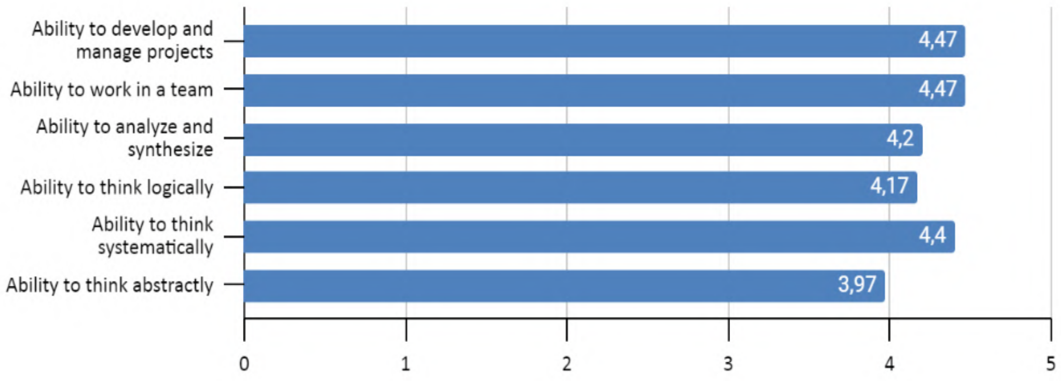

Figure 9: Bar chart visualizing the answers of respondents to the questions “Evaluate the impact of end-to-end design in the process of teaching programming on the formation and development of students’ competencies (1 - no impact, 5 - very significant)”.

Experimental teaching of students using the proposed methodology was conducted at Ternopil Volodymyr Hnatiuk National Pedagogical University and Mykhailo Dragomanov Ukrainian State University. Students studied selective subjects during which they studied the process of creating a computer game using the end-to-end design method. Experimental training took place in the 3-4th year and was the final stage of learning programming by future IT professionals.

During the course, the lecturer covered all aspects of using C# to design a game in the Unity environment throughout the duration of the course. At the same time, during the practical classes, students also performed end-to-end design of their game. After completing the course, students were surveyed to get feedback. This feedback made it possible to make the latest improvements and ensure that the TopDownShooter game would be interesting and enjoyable for players. According to the students surveyed (figure 10), the balance of game difficulty was optimal.

While working on developing a methodology for learning the C# programming language and using it to create game projects in Unity, we needed to determine what programming knowledge and skills are required for game development. According to the students, the required skills are: keyboard and mouse input processing (character movement, shooting), scripting skills, knowledge of C# syntax, and the ability to implement an event system (figure 11).

An experimental check of the effectiveness of the proposed teaching methodology confirmed its feasibility and practical value. The participants of the educational process were able to master the basics of C# programming, get acquainted with the features of the Unity environment, and create game projects on their own. The results of the experiment showed that the integration of gaming technologies into teaching helps to increase student motivation, develop their technical and creative skills, and improve their learning. The proposed methodology has proven to be effective in combining theoretical knowledge with practical experience, which is key to training competent IT professionals.

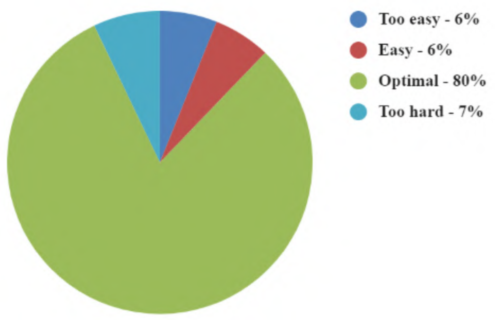

Figure 10: Circular diagram visualizing the answers of students to the questions "How would you rate the game's balance of difficulty?"

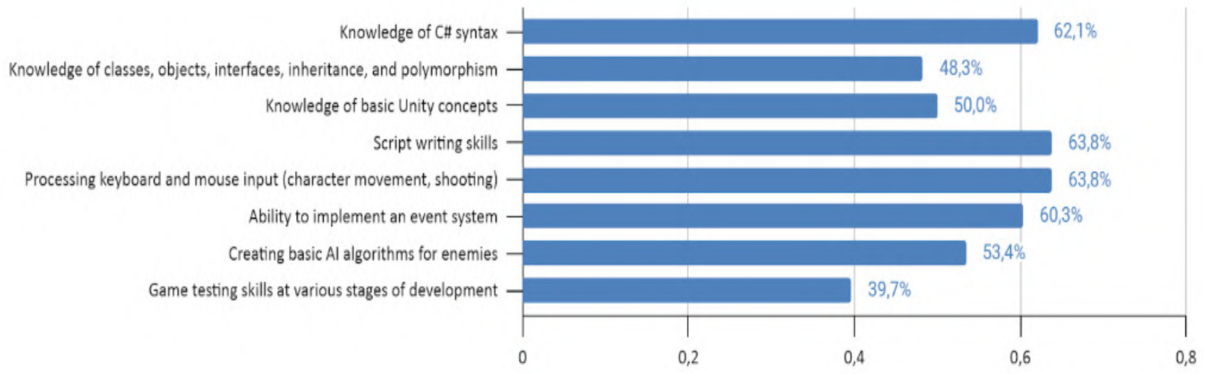

Figure 11: Bar chart visualizing the answers of students to the questions "What programming knowledge do you think you need to have to develop such a game?"

5. Conclusion

The level of development of modern information technologies and their penetration into all spheres of human activity determine the requirements for training of modern information technology specialists. The training of such specialists should be carried out in accordance with the level of technology development and ensure the formation of professional and key competencies in university graduates. Programming plays a key role in the structure of professional training of future information technology specialists. In our study, we proposed a methodology for teaching programming based on end-to-end design. The expediency of using this approach can be explained by the need to develop students' teamwork skills, the ability to implement projects in future professional activities, and an understanding of the software design process from the idea of creation to the stage of its release and subsequent support.

The study proposed the idea of teaching students to design a computer game as part of an elective course. To create a computer game, we consider it appropriate to use the Unity environment, as it is one of the most popular engines for creating computer games. Its use creates conditions for students' interest in game development, and allows us to demonstrate the possibilities of using modern technologies in the video industry. The developed course content includes the following topics: " Unity Overview", "Unity C# Programming Basics", "Designing and Creating Simple Game Mechanics", " Completing a Project". It is designed for six months of study, and therefore allows you to master the skills of end-to-end software design in a short time.

The end-to-end design method aims to teach students to work on complex, multi-stage tasks in which they must think through solutions not only for individual stages, but also for how these solutions will fit into the overall architecture and life cycle of the final product. Therefore, students' mastery of programming based on end-to-end software design contributes to a deeper understanding of not only programming itself, but also the peculiarities of all stages of software development, implementation and maintenance throughout its life cycle. Several stages are important for organizing the teaching of programming in the process of end-to-end design. At the preparatory stage, the lecturer must prepare a sufficient number of areas (or names) of software tools to be developed, clearly describe possible programming environments, languages, and technologies, prepare the content of the lecture material, and assign tasks to the project stages. In lectures, the teacher presents some theoretical material and demonstrates the application of this material to a specific stage of software design. In practical classes, students work together in a group setting to work on the design stages of a selected software tool (in our study, a computer game).

The survey of teachers found that 43.3% of teachers develop one large project during the programming course, and only 10% develop computer games during the course. However, developing one large project does not mean end-to-end software development. It can be one large project that is part of a final software product. Most lecturers prefer students to complete individual small tasks (90% of respondents) or to complete small group projects as part of educational practice (73.3%) or as part of a particular subject (56.7%). Respondents rate the impact of cross-cutting design on the effectiveness of programming education quite highly (with a score of 3.45 out of 5). Moreover, the results of the survey show that most of the programming teaching methods we have selected were highly rated Figure 8). Lecturers highly appreciated the impact of the end-to-end design methodology in teaching programming on the development of students' systematic, logical, and abstract thinking, the formation of teamwork skills, communication skills, and the ability to develop and manage projects.

Thus, organized training is an effective propaedeutic before students' practical training. The survey of respondents confirmed that it is the development of computer games and web applications (33.3% each) that arouse the greatest interest among students. The overwhelming majority of respondents noted that the use of this methodology is most conducive to the development of such competencies as teamwork and project development and management, which are important for future professional activities along with actual programming knowledge. The positive aspect of this course design and teaching is its practical focus on mastering programming language tools, and on the other hand, on implementing projects that require independent search and study of additional material. By mastering all stages of software development, students develop an understanding of software as a system.

Author Contributions

Conceptualization, Ihor A. Tverdohlib; methodology, Ihor A. Tverdohlib, Svitlana O. Leshchuk and Tetiana V. Pidhorna; validation, Tetiana V.Pidhorna; software, Denys I. Huska and Albina A.Patyashina; writing – original draft preparation, Svitlana O. Leshchuk and Denys I. Huska; writing—review and editing, Ihor A. Tverdohlib. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Data Availability Statement

No new data were created or analysed during this study. Data sharing is not applicable.

Conflicts of Interest

The authors declare no conflict of interest.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Y. S. Ramskyi, Y. O. B., I. A. Tverdokhlib, A. Y. Ramskyi, The use of open online courses in a blended learning environment in the training of future information technology specialists, Information Technologies and Learning Tools 84(4) (2021) 138–157. doi:10.33407/itlt.v84i4.4431.

[2] A. M. Striuk, S. O. Semerikov, Professional competencies of future software engineers in the software design: teaching techniques, Journal of Physics: Conference Series 2288 (2022) 012012. doi:10.1088/1742-6596/2288/1/012012.

[3] A. M. Striuk, S. O. Semerikov, H. M. Shalatska, V. P. Holiver, Software requirements engineering training: problematic questions, in: Proceedings of the 4th Workshop for Young Scientists in Computer Science & Software Engineering (CS&SE@SW 2021), volume 3077, 2022, p. 3 – 11. URL: https://ceur-ws.org/Vol-3077/paper01.pdf.

[4] S. Papadakis, A. E. Kiv, H. M. Kravtsov, V. V. Osadchyi, M. V. Marienko, O. P. Pinchuk, M. P. Shyshkina, O. M. Sokolyuk, I. S. Mintii, T. A. Vakaliuk, A. M. Striuk, S. O. Semerikov, Revolutionizing education: using computer simulation and cloud-based smart technology to facilitate successful open learning, in: S. Papadakis (Ed.), Joint Proceedings of the 10th Illia O. Teplytskyi Workshop on Computer Simulation in Education, and Workshop on Cloud-based Smart Technologies for Open Education (CoSinEi and CSTOE 2022) co-located with ACNS Conference on Cloud and Immersive Technologies in Education (CITEd 2022), Kyiv, Ukraine, December 22, 2022, volume 3358 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 1–18. URL: https://ceur-ws.org/Vol-3358/paper00.pdf.

[5] E. Vidal, R. Gacitúa, M. Diéguez, C. Cachero, Correspondence analysis between programming teaching approaches, in: XIV Jornadas Iberoamericanas de Ingenieria de Software e Ingenieria del Conocimiento, JIISIC 2019, Escuela Superior Politecnica de Chimborazo, 2019, pp. 177–188.

[6] S. O. Leshchuk, Y. S. Ramskyi, A. V. Kotyk, S. V. Kutsiy, Design a progressive web application to support student learning, in: A. E. Kiv, S. O. Semerikov, V. N. Soloviev, A. M. Striuk (Eds.), Proceedings of the 4th Workshop for Young Scientists in Computer Science & Software Engineering (CS&SE@SW 2021), Virtual Event, Kryvyi Rih, Ukraine, December 18, 2021, volume 3077 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 83–96. URL: https://ceur-ws.org/Vol-3077/paper16.pdf.

[7] O. G. Glazunova, O. V. Parhomenko, V. I. Korolchuk, T. V. Voloshyna, The effectiveness of GitHub cloud services for implementing a programming training project: students' point of view, Journal of Physics: Conference Series 1840 (2021) 012030. doi:10.1088/1742-6596/1840/1/012030.

[8] I. A. Tverdokhlib, O. V. Klochko, O. A. Sharyhin, V. M. Fedorets, Collaborative learning in the system of training future information technologies specialists as an educational strategy for the fundamentalization of the sustainable development of education, in: S. O. Semerikov, A. M. Striuk, M. V. Marienko, O. P. Pinchuk (Eds.), Proceedings of the 7th International Workshop on Augmented Reality in Education (AREdu 2024), volume 3918, CEUR-WS.org, 2024, pp. 206–225. URL: https://ceur-ws.org/Vol-3918/paper131.pdf.

[9] S. Korom, Z. Illés, Systemic Thinking in Programming Education, in: P. K. Singh, Y. Singh, J. K. Chhabra, Z. Illés, C. Verma (Eds.), Recent Innovations in Computing, Springer Singapore, Singapore, 2022, pp. 645–657. doi:10.1007/978-981-16-8892-8_49.

[10] R. D. Arnold, J. P. Wade, A Definition of Systems Thinking: A Systems Approach, Procedia Computer Science 44 (2015) 669–678. doi:10.1016/j.procs.2015.03.050.

[11] N.-G. Barcan, A. Alexandrescu, T. Turcanu, Gamification of the Learning Process for Acquiring Logical Thinking in Programming, in: 2024 23rd RoEduNet Conference: Networking in Education and Research (RoEduNet), 2024, pp. 1–6. doi:10.1109/RoEduNet64292.2024.10722314.

[12] N. Balyk, I. Grod, Y. Vasylenko, V. Oleksiuk, Y. Rogovchenko, Project-based learning in a computer modelling course, Journal of Physics: Conference Series 1840 (2021) 012032. doi:10.1088/1742-6596/1840/1/012032.

[13] R. Ruslan, L. Lu'mu, M. M. Fakhri, A. Ahmar, D. Fadhilatunisa, Effectiveness of the Flipped Project-Based Learning Model Based on Moodle LMS to Improve Student Communication and Problem-Solving Skills in Learning Programming, Education Sciences 14 (2024) 1021. doi:10.3390/educsci14091021.

[14] Z. Nurbekova, T. Tolganbaiuly, P. Tazabekova, G. Abildinova, B. Nurbekov, Enhance Students' Motivation to Learn Programming Through Projects, International Journal of Emerging Technologies in Learning (iJET) 15 (2020) pp. 133–144. doi:10.3991/ijet.v15i21.16537.

[15] Glazunova, Olena, Voloshyna, Tetiana, Korolchuk, Valentyna, Parhomenko, Oleksandra, Cloud-oriented environment for flipped learning of the future it specialists, E3S Web Conf. 166 (2020) 10014. doi:10.1051/e3sconf/202016610014.

[16] P. Srivastava, M. Kang, S. K. Gonugondla, S. Lim, J. Choi, V. Adve, N. S. Kim, N. Shanbhag, PROMISE: An End-to-End Design of a Programmable Mixed-Signal Accelerator for Machine-Learning Algorithms, in: 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), 2018, pp. 43–56. doi:10.1109/ISCA.2018.00015.

[17] K. Winkle, E. Senft, S. Lemaignan, LEADOR: A Method for End-To-End Participatory Design of Autonomous Social Robots, Frontiers in Robotics and AI 8 (2021). doi:10.3389/frobt.2021.704119.

[18] C. Orhei, S. Vert, M. Mocofan, R. Vasiu, End-To-End Computer Vision Framework: An Open-Source Platform for Research and Education, Sensors 21 (2021). URL: https://www.mdpi.com/1424-8220/21/11/3691. doi:10.3390/s21113691.

[19] N. M. Nguyen, N. Ray, End-to-end Learning of Convolutional Neural Net and Dynamic Programming for Left Ventricle Segmentation, 2019. URL: https://arxiv.org/abs/1812.00328. arXiv:1812.00328.

[20] D. Magda, C. Gardner-McCune, A. Kulkarni, Y. Jimenez, S. Chu, Supporting End-to-End Coding and Use of Arduinos in a Formal Classroom Environment , in: 2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), IEEE Computer Society, Los Alamitos, CA, USA, 2023, pp. 184–188. doi:10.1109/VL-HCC57772.2023.00030.

[21] S. S. Korniienko, P. V. Zahorodko, A. M. Striuk, A. I. Kupin, S. O. Semerikov, A systematic review of gamification in software engineering education, in: S. O. Semerikov, A. M. Striuk (Eds.), Proceedings of the 6th International Workshop on Augmented Reality in Education (AREdu 2023), volume 3844, CEUR-WS.org, 2023, pp. 83–95. URL: https://ceur-ws.org/Vol-3844/paper04.pdf.

[22] M. Frydenberg, Creating Games as Authentic Learning in the Information Technology Classroom., International Association for Development of the Information Society (2015). URL: https://api.semanticscholar.org/CorpusID:62978112.

[23] L. Végh, V. Stoffová, Learning object-oriented programming by creating games, eLearning and Software for Education (2019). URL: https://api.semanticscholar.org/CorpusID:257195841.

[24] Google Sheets: Online Spreadsheets & Templates | Google Workspace — workspace.google.com, https://workspace.google.com/products/sheets/, 2024. [Accesssed 20-12-2024].

[25] D. Kolb, Experiential Learning: Experience As The Source Of Learning And Development, volume 1, 1984. URL: https://www.researchgate.net/publication/235701029_Experiential_Learning_Experience_As_The_Source_Of_Learning_And_Development.

[26] J. Sweller, J. J. G. Van Merrienboer, F. Paas, Cognitive Architecture and Instructional Design: 20 Years Later, Educational Psychology Review 31 (2019) 261–292. doi:10.1007/s10648-019-09465-5.

[27] G. Dogara, S. Saud, Y. Kamin, M. Hamid, M. Nordin, Developing Soft Skills through Project Based Learning in Technical and Vocational Institutions, International Journal of Engineering and Advanced Technology 9 (2019) 2842–2847. doi:10.35940/ijeat.A9803.109119.

[28] End-to-End Software Development Services: Process & Benefits — integrio.net, https://integrio.net/blog/end-to-end-software-development-services, 2024. [Accessed 20-12-2024].

[29] J. Socha, P. Norton, Assembly Language for the PC, Brady programming library, Brady, 1992. URL: https://books.google.com.ua/books?id=NqZ9QgAACAAJ.
[30] H. Schildt, Java: A Beginner's Guide, Ninth Edition, McGraw Hill LLC, 2022. URL: https://books.google.com.ua/books?id=YGJVEAAAQBAJ.
[31] Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine — unity.com, https://unity.com/, 2024.