Informatics in Education, 2023, Vol. 22, No. 2, 257–276

DOI: 10.15388/infedu.2023.13

Help Students Learn Interpreted Petri Nets with Minecraft

Iwona GROBELNA, Małgorzata MAZURKIEWICZ, Damian JANUS
Faculty of Computer, Electrical and Control Engineering, University of Zielona Góra, Poland
e-mail: i.grobelna@iee.uz.zgora.pl, m.mazurkiewicz@issi.uz.zgora.pl, 104702@stud.uz.zgora.pl

Received: January 2022

Abstract. *Background:* Petri nets are a formal specification technique for modelling of control processes and modern flexible manufacturing systems. Interpreted Petri nets take into account input and output signals, allowing to apply them in any control system or even in control part of a cyber-physical system. Due to the fact that Petri nets are not used in the industrial practice, the students sometimes lack motivation to learn them. *Contributions:* In the paper we propose how to help students learn interpreted Petri nets with Minecraft (as a game-based learning). We show how interpreted Petri nets can be modelled in Minecraft and how they communicate with the surrounding environment via input and output signals to visualize control processes. The proposed approach has been validated experimentally among university students. *Hypotheses*: (1) Creating interpreted Petri net models with Minecraft helps to understand the basic principles; (2) Minecraft makes the course more attractive. *Methodology:* Students were divided into an experimental group (with game-based learning) and a control group (with traditional learning). The experimental group filled in a knowledge test twice (on the entry and on the exit) and a questionnaire. The control group filled in the same knowledge test at the end of the course. *Findings:* The observations confirm that the Minecraft-based teaching of interpreted Petri nets allows to gain better results in final tests, making at the same time the course more attractive and enjoyable.

Keywords: control systems, education, game-based learning, Petri nets.

1. Introduction

Petri nets are a general mathematical formalism for representation of discrete-event systems (Silva, 2013). They are widely used for specification of control (Giua and Silva, 2018) and manufacturing systems (Grobelna and Karatkevich, 2021), mainly because of the rich support by various analysis and verification methods (Karatkevich, 2007; Giua and Silva, 2017). Their graphical representation is also quite simple – basic elements include places (drawn as circles), transitions (drawn as bars), arcs (connecting places and transitions, or vice-versa, with each other) and tokens (drawn as dots inside places).

Despite the very limited set of elements, it is possible to reflect such behavior as choice, concurrency or resource sharing. Interpreted Petri nets (Grobelna *et al.*, 2019), as a special type of ordinary Petri nets, take into account additionally the presence of input and output signals to communicate with the environment. The appropriate structure of the created Petri net together with signal assignments allows to describe the functionality of a designed control system. Therefore, it is important to understand the interpretation and behavior of the model, in order to correctly establish a formal specification.

The process of teaching students the concepts of interpreted Petri net may be supported by modern technology. Students all over the world face nowadays the problem of remote education (Ali, 2020; Bond *et al.*, 2021; Toquero, 2021) and the theory is sometimes difficult to be used in practice. There are some approaches that may enrich the learning process, mainly by using some dedicated tools or other user-friendly forms, like interactive games or applications. In our previous paper, we proposed a Scratch-based method of requirements definition for control systems verification (Grobelna, 2020). That study showed that when using Scratch – it is easier for the students to define the requirements, in comparison to conventional temporal logic formulas. Since that study, the course of Discrete Control Systems at the University of Zielona Góra (Poland) is enriched with the visual programming language Scratch. This has also motivated us to continue the research and find out new ways of how to better inspire and help students learn the theory and practice of Petri nets. This time, we have chosen game-based learning (Plass *et al.*, 2015; Coleman and Money, 2020) and the Minecraft game (Cipollone *et al.*, 2014; Coleman and Money, 2020), that has so far proved to be successful in other domains of education (Baek *et al.*, 2020).

Minecraft is an open-world survival game with no specific goal of the game. Players have unlimited possibilities to explore and modify the randomly generated game world. The main idea behind Minecraft is to build structures consisting of 3D blocks. Minecraft has been continuously enjoying unflagging popularity for over 12 years. The popularity, availability, as well as unlimited freedom and the ease of creating even a very sophisticated game world, made Minecraft used in various areas of education. The educational potential of Minecraft can be successfully integrated with various school subjects (Youngkyun *et al.*, 2020).

In the paper we discuss how the process of teaching interpreted Petri nets may be supported by playing a game. In particular, we show how to model an interpreted Petri net in Minecraft and how to use the possibilities of Minecraft to connect it with a control process. We performed the study on 51 students in the courses of Discrete Control Systems and Modelling of Cyber-Physical Systems at the University of Zielona Góra (Poland) in order to evaluate the proposed approach. The enrolled students were divided into an experimental group with game-based learning and a control group with traditional learning. The control group trained the concepts taught previously in the lecture class using conventional methods, the experimental group used additionally the Minecraft game.

Before starting our experimental study, we have defined the two hypotheses:

(1) *Creating interpreted Petri net models with Minecraft helps to understand the basic principles* and
(2) *Minecraft makes the course more attractive*.

In order to test the first hypothesis, we have performed the single-choice knowledge test – before and after using Minecraft – and compared the obtained results. The control group filled in the same test at the end of the course, we have also compared these results with the experimental group. In order to test the second hypothesis, we have asked the students to express their feelings just after the experiment.

The main contributions of the paper are summarized as follows:

- We introduce a game-based approach to the area of control system education.
- We show how interpreted Petri net models can be visualized in Minecraft.
- We illustrate how interpreted Petri net models correspond to visualization of control processes via signals.
- We describe the experimental study among students that confirms both our hypotheses.
- We discuss the detailed results obtained in the study.

The remainder of the paper is structured as follows. Section 2 presents the state of the art regarding the theory of Petri nets, various approaches that contribute to the learning process, game-based learning and the recent educational approaches using Minecraft. Section 3 proposes the novel modelling approach with interpreted Petri nets. Section 4 describes the conducted study. Section 5 discusses the results of the experiment. Finally, section 6 summarizes and concludes the paper.

2. State of the Art

2.1. *Basic Theory on (Interpreted) Petri Nets*

Let us formally introduce the concept of Petri nets and, in particular, interpreted Petri nets and some basic properties and behaviors.

Definition 1. A *Petri net* (Murata, 1989) is a four-tuple $PN = (P, T, F, M_0)$, where P is a finite set of places, T is a finite set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs and M_0 is an initial marking. A *marking* involves all places that contain a token. A transition is *enabled*, if each of its input places contain a token. A transition can be *fired*, if it is enabled. Then, a token is removed from all its input places and added to all its output places. A marking is *reachable* from any other marking, if it can be reached by a sequence of transition firings.

In a Petri net, places may be connected via arcs only with transitions, and vice versa. However, it is possible to model various behaviors by using the appropriate structure. Basic structures in Petri nets are shown in Fig. 1.

Definition 2. A Petri net is *live* (David and Alla, 2010), if from any reachable marking it is possible to fire any transition by a sequence of firings of other transitions.

Definition 3. A Petri net is *safe* (David and Alla, 2010), if there is no reachable marking such that the place contains more than one token.

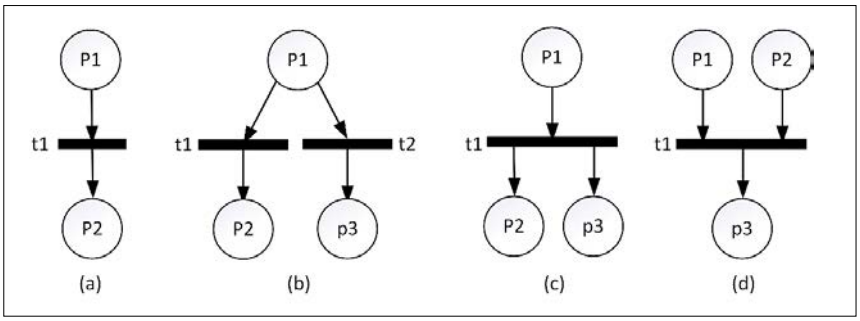

Fig. 1. Basic structures in Petri nets:
(a) sequence, (b) choice, (c) concurrency and (d) synchronization.

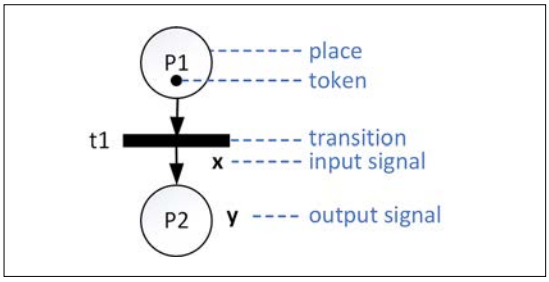

Fig. 2. Elements of an interpreted Petri net.

Definition 4. An *interpreted Petri net* (Grobelna *et al.*, 2019) is a six-tuple $IN = (P, T, F, M_0, X, Y)$, where the first four elements describe a Petri net, that is live and safe, X is a finite set of logic input signals, Y is a finite set of logic output signals. A transition in an interpreted net can be fired, if it is enabled and all the conditions of its input signals (assigned to it) are fulfilled.

The elements of an interpreted Petri net are shown in Fig. 2. It should be noted that the structure of ordinary Petri nets is here supplemented by input and output signals. Input signals are assigned to transitions as their guards, while output signals are assigned to appropriate places. And so, in order to fire a transition, its input places must contain a token and the guard (specified with input signals) must be satisfied. In the example, in order to fire transition *t1*, place *P1* has to contain a token, and additionally input signal *x* must be active. As a result, a token is removed from place *P1* and added to place *P2*.

2.2. *Learning Petri nets*

The teaching process of Petri nets considers the combined areas of control systems and mathematics (computer logic). There are some Petri net tools (Thong and Ameedeen,

2015), that enable drawing the nets and also their basic analysis, like Yasper (van Hee *et al.*, 2006), CPN Tools (Yu *et al.*, 2018), PIPE2 (Dingle *et al.*, 2009), GPenSIM (Davidrajuh *et al.*, 2018), or IOPT-tools (Gomes *et al.*, 2013). These tools are used mainly for checking some basic properties of the Petri nets, like boundedness, safeness or deadlock freedom. There are also some software tools dedicated for education, like P3-PN (Gašević and Devedžić, 2004), PetriDotNet (Vörös *et al.*, 2018), GreatTeach (Amparore and Donatelli, 2018) or PN2ARDUINO (Kučera *et al.*, 2019). These tools have been proposed especially for improving the learning process of students. The available tools, mostly developed by scientific institutions, focus in particular on Petri nets or their analysis and provide poor user-experience. For effective usage of them, it is assumed that the user is already familiar with the basic theory of Petri nets.

The current challenges regarding Petri nets education are identified in (Grobelna and Karatkevich, 2021), where additionally, also the summary of trends in industrial electronic education and remote laboratories, as well as some approaches to teaching Petri nets are provided. Unfortunately, there are just a few papers focusing on teaching Petri nets in detail. Š. Korečko (Korečko, 2018) proposes to use interactive model building with active audience participation. It has been proved suitable for short and intensive courses. The same author describes and evaluates also in (Korečko, 2019) the implementation of project-based learning to a graduate course focusing on the event-driven simulation. In general, project-based learning as a student-centred pedagogy (McCabe and O'Connor, 2014), is nowadays gaining more attention in engineering education (Reis *et al.*, 2017). Schmitz *et al.* show in (Schmitz *et al.*, 2016) how to utilize Petri nets for teaching practical courses on collaborative software engineering (e.g. by improving the worksheets). The results of practical experiments on students are also presented, with a conclusion that the students worked more independently with less confusion and frustration.

The recent challenges in teaching Petri nets (Grobelna and Karatkevich, 2021) clearly indicate the conditions of remote education and the problem of how to fit interdisciplinary aspects into the limited duration of a course. Having in mind these challenges, we propose to enrich the course at the university with game-based learning. The solution may be realized by the students remotely (each one working at home). Moreover, the willingness to play games (in particular Minecraft) is usually high enough to spend some more time than actually included in the course.

2.3. *Game-Based Learning*

The use of computer games in education is not a new idea. The first applications of this type were created in 1967. An example is the Logo Programming software, which has been used in education and is still used, among others, for learning programming and mathematics (Walsh, 2019; Salomon *et al.*, 2020). Salomon *et al.* point out that Logo Programming has become a symbol of change in basic mathematics education as well as in the very nature of the school. The Logo encourages children to reflect their own relationships with science and motivates teachers to rethink their relationships with the curriculum and with children.

In the recent work (Panja and Berge, 2021) factors that should be considered when creating a game-based educational model are discussed. It has been indicated that not all students/players engage with Minecraft in the same way, and that prior game experience influences how the student learns (acquires knowledge), which affects the results.

In another paper (López-Fernández *et al.*, 2021), the authors present evidence of the effectiveness of game-based learning in computer science education when using educational video games created by teachers using authoring tools. The results show that game-based learning is as effective as traditional learning in terms of gaining knowledge, but it was much more effective in increasing students' motivation. Students who learned by playing educational video games found the experience more motivating and fun. The vast majority of them preferred the game-based approach to traditional teaching.

Çağlar in the work (Çağlar, 2020) showed that the influence of computer games on programming education was significant and made several recommendations for both teachers and students. In (Chen *et al.*, 2020) the effects of competition on game-based learning are described. The authors found that competition in game learning was effective for math, science, and language, but not for social sciences and other subjects.

Taub *et al.* (Taub *et al.*, 2020) demonstrated that the moderate degree of agency provided to learners in game-based learning environments leads to better learning outcomes without sacrifice and without negative emotional experience, showing how even low levels of agency can positively impact learning, problem solving, and affect during game-based learning.

2.4. *Recent Approaches of Minecraft in Education*

The educational potential of Minecraft has been widely described in the literature for many years. The game is used, among others, to learn science and technology. Initial research into the suitability of Minecraft for learning chemistry is described in (Panja and Berge, 2021). The factors that should be considered when building a Minecraft-based learning model are discussed. Based on the obtained results, a preliminary conclusion was formulated that the game mechanics can provide commitment and effective learning depending on how the student learns. At the same time, the need for further research in this area was indicated.

The possibility of using Minecraft to learn math is discussed in (Moore, 2018; Jensen *et al.*, 2020). The articles describe how the game can help students understand geometry and measurement, as well as calculate areas and volumes.

Other papers (Prayaga *et al.*, 2016; Jensen *et al.*, 2020) show that Minecraft can also be successfully used in teaching computer science and digital logic. In these articles, the authors describe their experiences in teaching programming and understanding basic logic gates.

Minecraft has been used in machine learning (Bezzubov and Frolov, 2021) as well as in teaching artificial intelligence (Singh, 2020). The article (Singh, 2020) describes the use of Minecraft as a platform for teaching artificial intelligence (AI) through project-based learning. The Minecraft platform was used as part of an open-ended undergradu-

ate course with 82 different projects covering topics in the areas of navigation, instruction following, object detection, combat, and music/image generation.

In the field of humanities, the educational potential of the game was used to teach reading and writing at the academic level (Ponce *et al.*, 2020), for developing collaboration, social and research skills (Mørch *et al.,* 2019; Hewett *et al.* 2020), in learning to plan and schedule (Master *et al.*, 2020) as well as in developing students' spatial abilities (Worsley and Bar-El, 2020; Carbonell-Carrera *et al.*, 2021).

As an environment with high intensity of use, Minecraft is also applied to conduct research, e.g., in the field of user behavioral biometrics (Siddiqui *et al.*, 2021), simulation of human demonstrations (Guss *et al.*, 2019) or photorealistic 3D block world synthesis (Hao *et al.*, 2021).

3. Modelling Interpreted Petri Net with Minecraft

3.1. *General Description*

In order to use Minecraft in the learning process of interpreted Petri nets it is necessary to understand how it can contribute to the education. The schematic presentation of the proposed idea is shown in Fig. 3. It is essential that both the Petri net models as well as the visualization of control processes can be realized in one game, introducing some animation to the theory.

On the left side, there is an interpreted Petri net, that describes the control process of two carriages movement. The structure of Petri net includes six places and five transi-

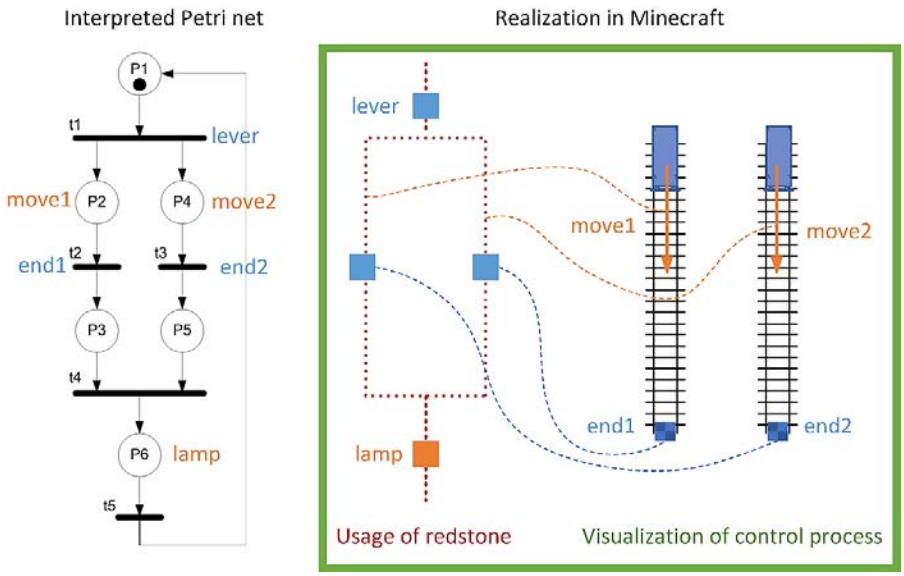

Fig. 3. Schematic presentation of the idea.

tions, three input and three output signals are used for communication with the environment. And so, after the signal *lever* becomes active, transition *t1* may be fired, and concurrent processes start. The output signals *move1* and *move2* force the carriages to move from their starting points to their ending points. The carriages move so long, until they reach their ending points (*end1* and *end2*, respectively). The adjusted structure of Petri net ensures that the system operates correctly, with no difference which carriage arrives first. Then, the two processes wait for each other to be synchronized, and finally, the *lamp* output signal becomes active. The structure of the interpreted Petri net is realized with redstone. The input signals may be related to levers or buttons, while the output signals to redstone lamps or other activators. Besides the Petri net model, it is also possible to model the physical plant that visualizes the control process. On the right side, a Minecraft realization is presented. The signals in the physical plant are connected with the structure of the interpreted Petri net, so the animation of a Petri net (token flow) results also in some attractive animations in the game.

3.2. *Modelling the Sequentiality*

The basic structure of a Petri net is the sequential ordering of places (see also Fig. 1a). Then, places and transitions appear one after the other. The modelling and realization in Minecraft are shown in Fig. 4. Each place has assigned an output signal (the simplest activator is a lightning element, *lamp1–lamp4* for places *P1–P4*). Input signals are assigned to transitions and correspond to the simple element of a lever. This allows the user to interactively switch the levers and enable the control flow. In a simple realization

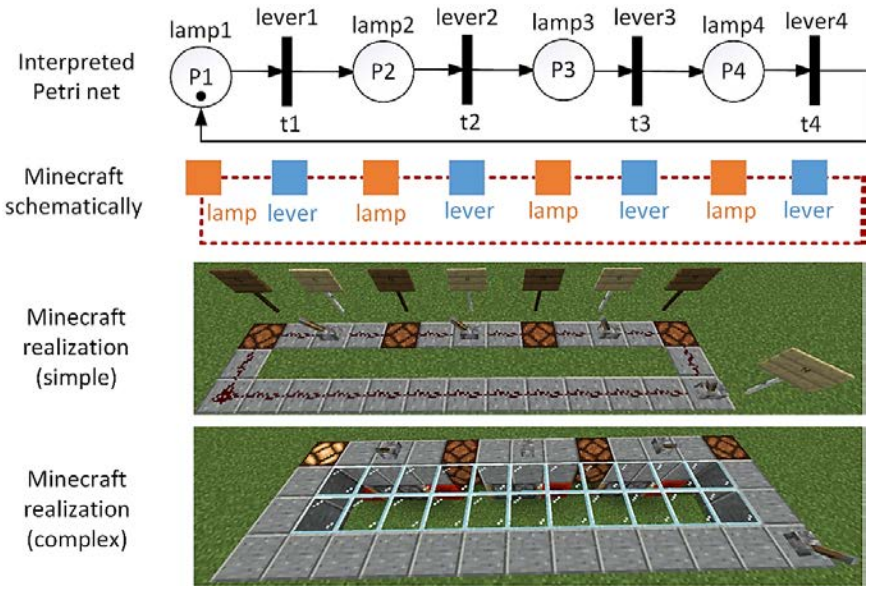

Fig. 4. Realization of sequentiality.

(using only redstone), the fired transition turns on both its input and outputs places, so it is necessary to follow the right direction. In a more complex realization (with additional piston elements), only the output places are turned on, so the pure animation (classic token flow) of an interpreted Petri net can be observed.

3.3. *Modelling the Choice*

Another frequently used construction is modelling of a choice (see also Fig. 1b). The modelling and realization in Minecraft are shown in Fig. 5. In the Petri net model it

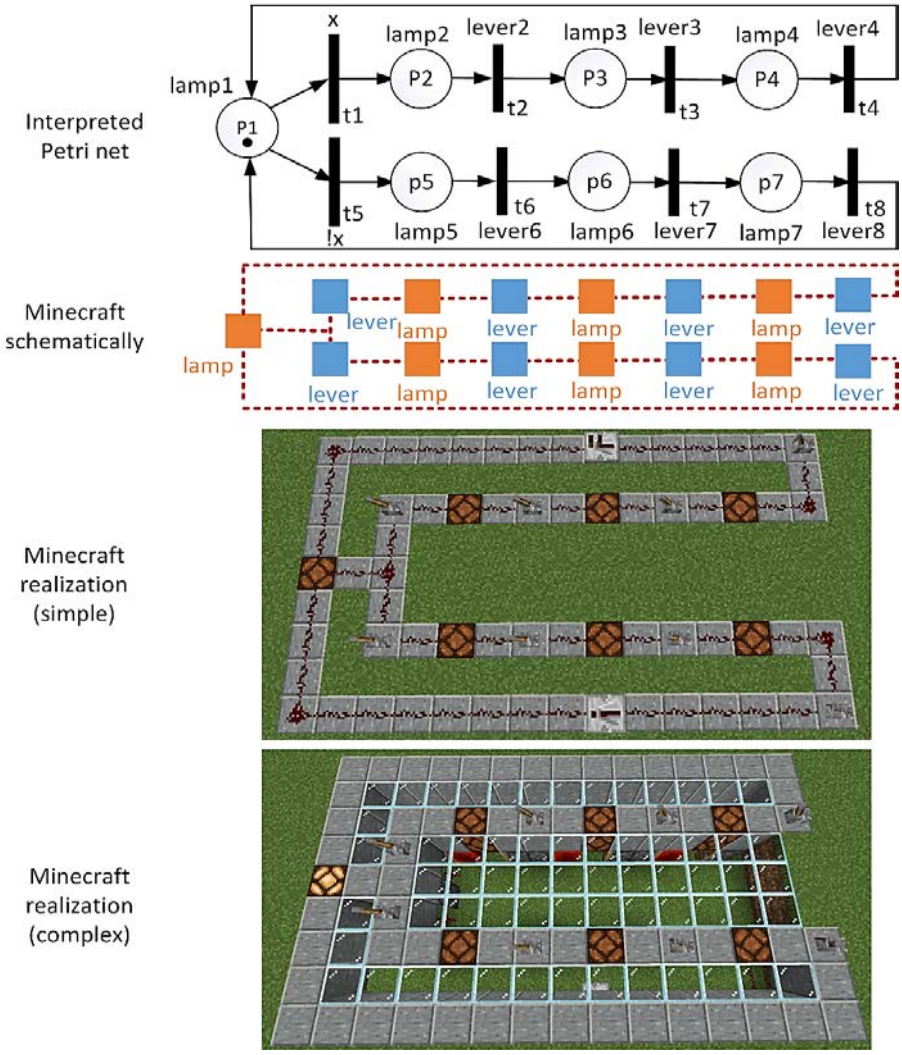

Fig. 5. Realization of choice.

is realized by connecting two transitions to one common input place (in the example: transitions *t1* and *t5*). In interpreted Petri nets, these transitions are usually guarded to ensure the determinism of the model (Lee, 2017; Wisniewski *et al.*, 2020). Therefore, it is important that the guards are mutually exclusive, e.g., *x* and *not x* (denoted as *!x*). Here also the places correspond to lamps in Minecraft, while the transitions to levers. Again, there are two possible realizations – a simpler one (suitable even for beginners) or a complex one (showing classic token flow).

3.4. *Modelling the Concurrency*

In control systems the behavior of concurrency (see also Fig. 1c and Fig. 1d) is of special interest. Indeed, most real industrial control processes are concurrent, so it is important that the concurrency is properly understood by students while attending the course. The realization of concurrency is illustrated in Fig. 6. In the structure of a Petri

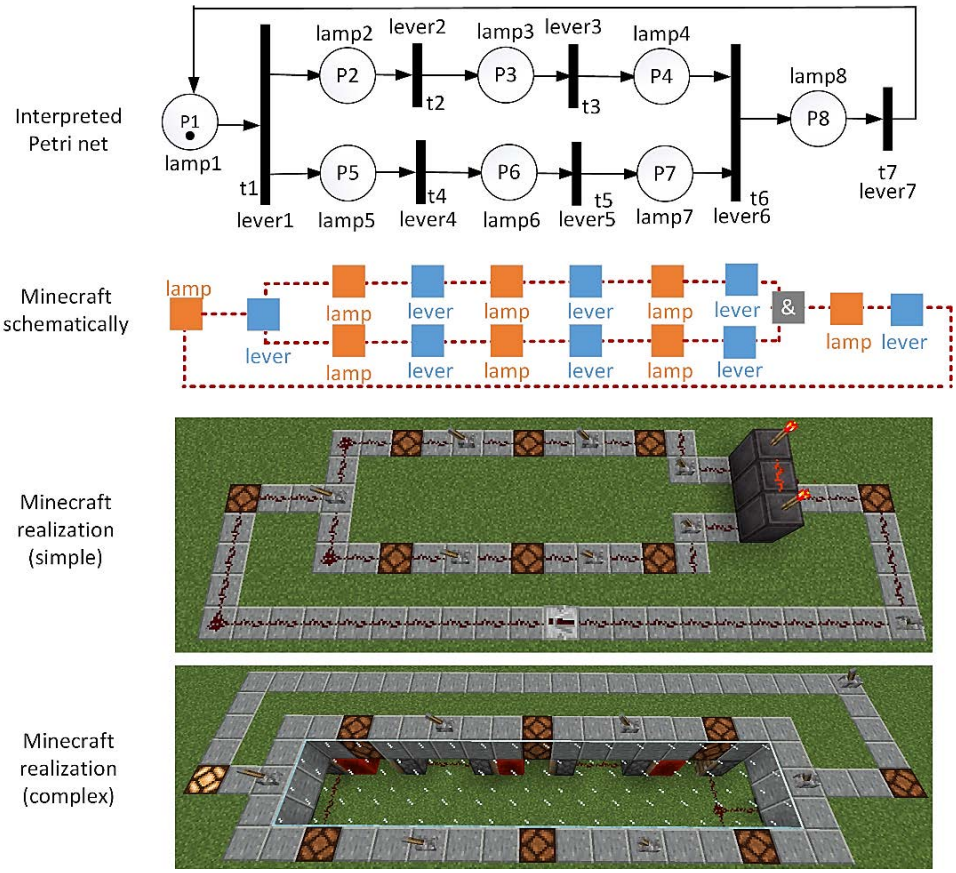

Fig. 6. Realization of concurrency.

net the begin of concurrency is annotated with a transition with one input place and two or more output places (in the example: transition *t1*), while the end of it (the synchronization) – with a transition with two or more input places and one output place (in the example: transition *t6*). The realization in Minecraft is more problematic, the usage of an AND gate is required to synchronize the processes (the flow proceeds if all processes come to the end, so all inputs to the AND gate are *TRUE*). Two possible realizations take place as before, in a complex one the synchronization mechanism is hidden underground.

3.5. *Connecting an Interpreted Petri Net Model with a Physical Plant*

Players of Minecraft spend much time on building various constructions, buildings and mechanisms (Cordeiro and Nelson, 2014). This ability can be used to visualize physical plants that realize some control processes. Short scenarios, which can be realized even by beginners, include controlling the movement of some carriages, their loading and unloading, or simple production processes.

The visualization of a control process seems therefore not to be a problem for Minecraft players. The main difficulty is how to correctly connect the particular elements of the process with the model of interpreted Petri net. The earlier realization of basic structures just with lamps and levers simplifies the understanding. Appropriate signals must then be connected with the corresponding actuators or sensors, e.g., leading from a redstone torch to a powered rail in order to move the carriage or from a button to a door (an element needed to enable token flow). Fig. 7 illustrates how the physical plant can be connected with the interpreted Petri net model. The scenario visualizes the control process of one carriage movement and goods transport.

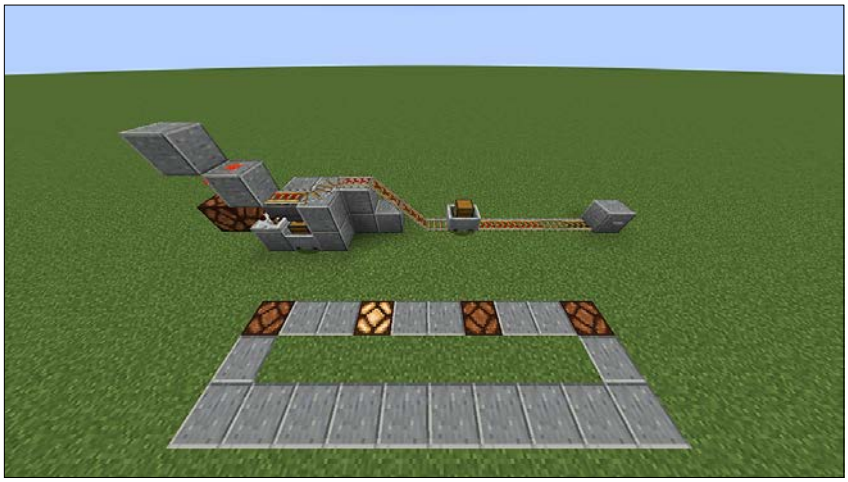

Fig. 7. Physical plant combined with interpreted Petri net model.

4. Conducted Study

In order to practically test our proposed approach of modelling interpreted Petri nets in Minecraft and evaluate its influence on the learning process, we have conducted a study among students in the courses of Discrete Control Systems and Modelling of Cyber-Physical Systems at the University of Zielona Góra (Poland). The study was performed in the academic year 2021/22 and involved 51 participants, all of them being males with an average age of around 20 years. The enrolled students were divided into an experimental group with game-based learning (43 students) and a control group with traditional learning (8 students). The disproportion results from the size of academic groups at the university. The participation in the experiment was entirely voluntary, the results had no effect on the final grades and all answers were anonymized (included no personal data enabling student's identification). All participants in the experiment have been playing Minecraft before and have known the basic rules regarding the game. Just before the experiment, they have also been introduced to the theory and practice of general Petri nets and interpreted Petri nets. The study was divided into four parts as shown in Fig. 8. The control group had instead traditional lessons and filled in the same single-choice knowledge test at the end of the course (referring to step 4 of the experiment).

4.1. *Part 1*

Prior to the experiment, the students were asked to take the test, which included both the theory and the practice of interpreted Petri nets. The test involved 15 questions with a single response (one correct answer out of four). After solving the test, the students were not informed about the correct answers. The time limit was set to 15 minutes (one minute per one question).

4.2. *Part 2*

The students were introduced to the proposed approach of modelling interpreted Petri nets in Minecraft (basic structures and their realizations as shown in Sections 3.2–3.4).

Fig. 8. Parts of the experimental study.

The presentation was followed by self-realization of the basic structures. Each student played his own game and tried to reconstruct the examples, focusing especially on the proper distinction between input and output signals. This part was realized within 90 minutes. Then, the students were asked to try out different structures of interpreted Petri nets supplemented by lamps and levers on their own during the time period of one week.

4.3. *Part 3*

The students were introduced to how to combine interpreted Petri nets with visualization of physical plants in Minecraft (introduction to scenario from Section 3.5). The presentation was followed by self-realization of a basic scenario with carriage movement. Then, the students were asked to realize some scenarios including interpreted Petri nets and visualizations of physical plants. This part could be realized either alone or together in a multiplayer mode. This part was realized within 90 minutes.

4.4. *Part 4*

After the experiment, the students were asked to take the same test again (with 15 single-choice questions) and fill in a questionnaire for rating the experience. In order to maintain the same conditions, the time limit for the test was set to 15 minutes (one minute per one question). The results of the tests could then be compared with each other to evaluate how the game-based learning influenced the learning process. Then, the students were asked to fill in a questionnaire (optional). They could also write some personal comments on the experimental way of learning. This has allowed us to get to know the subjective opinions of the students.

5. Presentation and Discussion of the Results

Our educational experiment has been performed on a group of 51 students. At the beginning, the experimental group filled in the test on the entry (part 1 of the study). Then, they took part in the game-based learning – after a short introduction they created the Petri net models by themselves (part 2), as well as some visualizations of control processes (part 3). At the end, they filled in the test on the exit, a questionnaire with their personal feelings and additional comments (part 4). This has allowed us to evaluate the proposed learning approach. The control group was provided with the same theory during the lectures, but instead of using Minecraft, the students were trained in the traditional way. At the end of the course, the control students filled in the same single-choice knowledge test as the experimental students to compare the results. Both groups were taught by the same academic teacher.

5.1. *Results of the Tests*

Both obligatory tests had 15 single-choice questions, so it was possible to achieve maximally 15 points. Comparing the results of the test on the entry and on the exit, we observed that the arithmetic average value has been increased by ca. 4 points (i.e. about 50%). At the same time, both the lowest value and the median value have increased by 5 points. The results are summarized in Table 1. The control group solved the test only once (at the end of the course). The results obtained by the experimental group were better than for the control group, with median value 12 (versus 11 for the control group) and average value 12,02 (versus 11,125 for the control group).

For the experimental group, we have also observed that the dispersion of values was much lower in the test on the exit, what means that the results obtained by the students were more similar than at the beginning. This shows that the students with worse outcomes have broadened their knowledge, and finally, the students achieved a similar level of Petri nets knowledge. Although these results alone are not surprising (more time spent on learning usually has a positive impact on the gained knowledge), the comparison

Table 1

Results of the test on the entry and on the exit (experimental group) versus control group

	Arithmetic average	The lowest value	Median value
Test on the entry	7,86	3	7
Test on the exit	12,02	8	12
Difference	+4,16 (+53%)	+5	+5
Control group	11,125	8	11

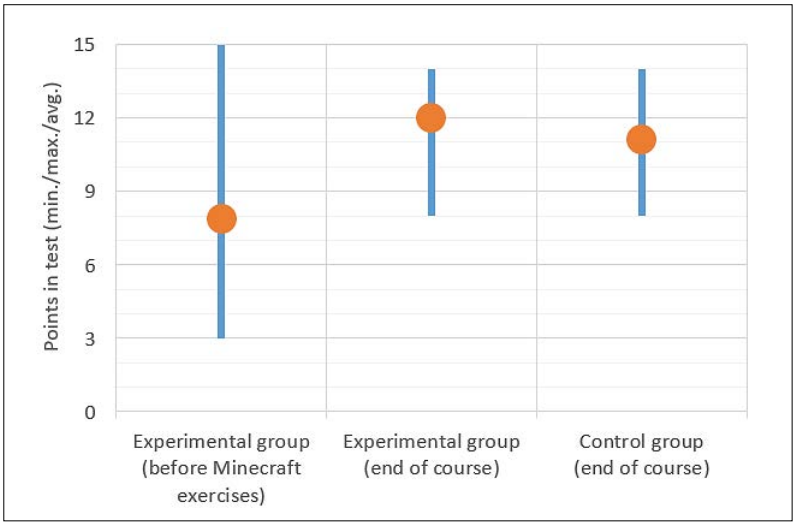

Fig. 9. Results comparison.

with the control group allows to observe the influence of game-based learning (chart in Fig. 9). The dispersion of values for the experimental group and the control group for the final knowledge test were the same, but the median and average values were better for the students who participated in Minecraft-based learning.

Based on these test results we can conclude that the knowledge of the students has been enriched with game-based learning. It also confirms the first hypothesis that "*Creating interpreted Petri net models with Minecraft helps to understand the basic principles*".

5.2. *Results of the Questionnaire*

Optionally, the participants filled in a questionnaire with their personal comments on the experimental way of learning. The questionnaire has been answered by 33 students. Each question has been rated in a scale from 1 to 5. The results are summarized in Table 2.

In general, we can conclude that the experimental learning of interpreted Petri nets was successful. The students found the lessons attractive and stated that they contributed to the better understanding of Petri nets (including token flow, modelling of concurrency and distinguishing between input/output signals). Therefore, the results of the questionnaire confirm both hypotheses: "*Creating interpreted Petri net models with Minecraft helps to understand the basic principles*" and *Minecraft makes the course more attractive"*.

In order to evaluate how the so-far experience in playing Minecraft influence the results, we have separately analyzed two groups of students (based on answers to question 5) – the first one that rated own skills as 1 to 3 points (17 students), the other one with skills rated as 4 or 5 points (16 students). The comparison is illustrated in Fig. 10.

We can conclude, that the general feelings were slightly better by students with higher Minecraft skills. However, the difference was not significant in comparison to the beginners. The opposite tendency has been observed only by question no 4, regarding

Table 2

Questions in the questionnaire. Note: all have been rated in the scale from 1 (little) to 5 (a lot)

ID	Question	Arithmetic average	Median value
1	How do you rate the attractiveness of Minecraft classes?	4,67	5
2	How has Minecraft contributed to a better understanding of token flow in Petri nets?	4,18	4
3	How has Minecraft contributed to a better understanding of concurrency modelling in Petri nets?	4,42	5
4	How has Minecraft contributed to a better distinguishing between input and output signals?	4,36	5
5	How do you rate your Minecraft skills?	3,48	3

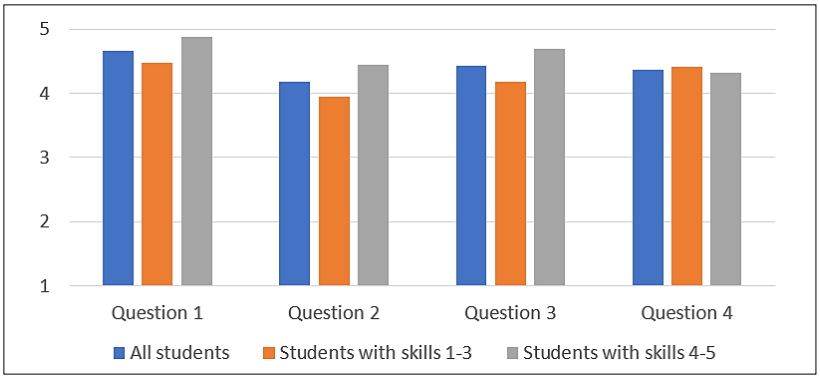

Fig. 10. Influence of the so-far Minecraft skills on the experiment.

the distinguishing between input and output signals. The reason may be that the more professional players had already worked with input and output signals, so they have known this aspect before, while for beginners it was a new experience.

5.3. *Comments on the Experiment*

We have also asked the students to write some additional comments (some selected opinions are listed in Table 3). We have observed that the game-based lessons provided much fun to the students and simplified the understanding. The possibility to create the Petri net models alone allowed to test various combinations and see the results on-the-fly. Moreover, the visualization of control processes itself was evaluated rather as game-playing (and not learning), also because of the possible realization in a multiplayer mode, where multiple students could create their plants together in one Minecraft world.

Table 3

Some opinions of the participants about the novel game-based learning approach

ID	Opinion
1	*Minecraft classes helped to improve the knowledge of Petri nets. During this type of classes, it is much easier to understand a given topic, because we create these nets by ourselves and it is a kind of practice.*
2	*Instead of imagining a Petri net in our head, we can easily construct it in the game and on-the-fly analyze the entire process, which is very helpful in case of a mistake. It will take less time to find the errors, because they will be immediately noticeable on implemented Petri net model. In order to use Minecraft as a "tool", we only need basic skills and knowledge of the game, and deepening these skills can greatly speed up and improve our work. Being a beginner will slow down our work rather than make it impossible.*
3	*Great idea to implement slightly abstract models (students' possibilities are limited to their imagination) and create Petri nets in Minecraft. This significantly increases the interest in the subject.*
4	*It is a very interesting and inspiring form of getting to know the functioning of a Petri net. A very good idea to combine "fun" with a very important issue.*
5	*A pleasant surprise, a much more pleasant form of teaching. I hope for more such surprises.*

6. Conclusions

In the article a novel approach to teaching interpreted Petri nets with Minecraft is proposed. The learning process has been extended with game playing, what allowed to introduce some fun into the standard academic course. Interpreted Petri nets, as a formal specification technique of control systems, are sometimes difficult to learn, mainly because of the mathematical background and differentiation of input and output signals. The limited duration of a course does not simplify the task of bringing the knowledge to the students. Game-based learning, adjusted to the age of students, is attractive enough to inspire them to spend more time on education.

Based on the realization of basic structures of interpreted Petri nets in Minecraft, together with visualizations of control processes, we have conducted an experiment among students to evaluate our idea (using an experimental group and a control group). We have formulated two hypotheses that have been confirmed: (1) *"Creating interpreted Petri net models with Minecraft helps to understand the basic principles"*; and (2) *"Minecraft makes the course more attractive"*. Our conclusions and observations are based on the results of a single-choice knowledge test (before and after the experimental way of studying and at the end of the course for the control group), as well as on the additional questionnaire with subjective feelings of the participants. The obtained results clearly show that enriching the standard course with a game-based learning increases the course attractiveness and inspires students to spend more time on "playing" with interpreted Petri nets. At the same time, it helps students learn interpreted Petri nets and gain better results in the knowledge test. The Minecraft-based learning was successful in increasing student motivation. Students who played Minecraft game, found this way of learning enjoyable and inspiring. The visualization of control processes was even evaluated rather as game-playing and not learning (multiple students could create their plants together in one Minecraft world).

Future research will continue the study by diversifying the exercises for students, so that they train not only how to construct a properly formed interpreted Petri net, but also how to fill in the missing part or how to correct a net which is not live or not safe.

Funding

The work is supported by The National Science Centre, Poland (grant number 2019/35/B/ST6/01683).

References

Ali, W. (2020). Online and remote learning in higher education institutes: A necessity in light of COVID-19 pandemic. *Higher Education Studies*, 10(3), 16–25.

Amparore, E.G., Donatelli, S. (2018). GreatTeach: A tool for teaching (stochastic) Petri nets. In: *Proceedings of the International Conference on Applications and Theory of Petri Nets and Concurrency*, Bratislava, Slovakia, Springer: Cham, Switzerland, 416–425.

Baek, Y., Min, E., Yun, S. (2020). Mining educational implications of Minecraft. *Computers in the Schools*, 37(1), 1–16.

Bezzubov, S.S., Frolov, A. A. (2021). Practical Use of Machine Learning in the Minecraft Computer Game. In: *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, 238–240, DOI: 10.1109/ElConRus51938.2021.9396731.

Bond, M., Bedenlier, S., Marín, V. I., Händel, M. (2021). Emergency remote teaching in higher education: mapping the first global online semester. *International Journal of Educational Technology in Higher Education*, 18(1), 1–24.

Çağlar, E. (2020). The Effect of Computer Game on Coding Learning: A Case Study of University Students. *Avrupa Bilim ve Teknoloji Dergisi*, 20, 456–465, DOI: 10.31590/ejosat.702460.

Carbonell-Carrera, C., Jaeger, A.J., Saorín, J.L., Melián, D., de la Torre-Cantero, J. (2021). Minecraft as a block building approach for developing spatial skills. *Entertainment Computing*, 38, 100427, DOI: 10.1016/j.entcom.2021.100427.

Chen, C., Shih, C., Law, V. (2020). The effects of competition in digital game-based learning (DGBL): a meta-analysis. *Educational Technology Research and Development*, 1–19.

Cipollone, M., Schifter, C.C., Moffat, R.A. (2014). Minecraft as a creative tool: A case study. *International Journal of Game-Based Learning (IJGBL)*, 4(2), 1–14.

Coleman, T.E., Money, A.G. (2020). Student-centred digital game–based learning: a conceptual framework and survey of the state of the art. *Higher Education*, 79(3), 415–457, DOI: 10.1007/s10734-019-00417-0.

Cordeiro, A., Nelson, E. (2014). *Minecraft Construction for Dummies*. John Wiley & Sons.

David, R., Alla, H. (2010). *Discrete, Continuous, and Hybrid Petri Nets*. Springer, Berlin.

Davidrajuh, R., Skolud, B., Krenczyk, D. (2018). Performance Evaluation of Discrete Event Systems with GPenSIM. *Computers*, 7(1):8.

Dingle, N.J., Knottenbelt, W.J., Suto, T. (2009). PIPE2: A tool for the performance evaluation of generalised stochastic Petri Nets. *ACM Sigmetr. Per 2009*, 36, 34–39.

Gašević, D., Devedžić, V. (2004). Teaching Petri nets using P3. *J. Educ. Technol. Soc.*, 7, 153–166.

Giua, A., Silva, M. (2017). Modeling, analysis and control of Discrete Event Systems: A Petri net perspective. *IFAC-PapersOnLine*, 50, 1772–1783.

Giua, A., Silva, M. (2018). Petri nets and Automatic Control: A historical perspective. *Annual Reviews in Control*, 45, 223–239.

Gomes, L., Moutinho, F., Pereira, F. (2013). IOPT-tools – A Web based tool framework for embedded systems controller development using Petri nets. In: *Proceedings of the 23rd International Conference on Field programmable Logic and Applications*, Porto, Portugal, 1–1.

Grobelna, I. (2020). Scratch-Based User-Friendly Requirements Definition for Formal Verification of Control Systems. *Informatics in Education*, 19(2), 223–238, DOI: 10.15388/infedu.2020.11.

Grobelna, I., Karatkevich, A. (2021). Challenges in Application of Petri Nets in Manufacturing Systems. *Electronics*, 10(18), 2305, DOI: 10.3390/electronics101 82305

Grobelna, I., Wiśniewski, R., Wojnakowski, M. (2019). Specification of Cyber-Physical Systems with the Application of Interpreted Nets. *IECON 2019 – 45th Annual Conference of the IEEE Industrial Electronics Society*. 5887–5891, DOI: 10.1109/IECON.2019.8926908.

Guss, W.H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M.M., Salakhutdinov, R. (2019). MineRL: A Large-Scale Dataset of Minecraft Demonstrations. *ArXiv*, abs/1907.13440.

Hao, Z., Mallya, A., Belongie, S.J., Liu, M. (2021). GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds. *ArXiv*, abs/2104.07659.

van Hee, K., Oanea, O., Post, R., Somers, L., van der Werf, Yasper, J.M. (2006). A tool for workflow modeling and analysis. In: *Proceedings of the 6th International Conference on Application of Concurrency to System Design*, Turku, Finland, 279–282.

Hewett, K., Zeng, G., Pletcher, B. (2020). The Acquisition of 21st-Century Skills Through Video Games: Minecraft Design Process Models and Their Web of Class Roles. *Simulation & Gaming*, 51, 104687812090497, DOI: 10.1177/104687812090 4976.

Jensen, E., Hanghøj, T. (2020). What's the math in Minecraft? A Design-Based Study of Students' Perspectives and Mathematical Experiences Across game and School Domains. *Electronic Journal of e-Learning*, 18, 261–274, DOI: 10.34190/EJEL. 20.18.3.005.

Karatkevich, A. (2007). *Dynamic Analysis of Petri Net-Based Discrete Systems*. Springer Science & Business Media: Berlin/Heidelberg, Germany.

Korečko, Š. (2018). Interactive Approach to Coloured Petri Nets Teaching. *Tech. Rep. IK-TR3*, Eötvös Loránd University, Faculty of Informatics, Budapest, Hungary.

Korečko, Š. (2019). Project-Based Approach to Teaching Event-Driven Simulation. In: *Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectron-*

ics (MIPRO), Opatija, Croatia, 1604–1608.
Kučera, E., Haffner, O., Leskovský, R. (2019). PN2ARDUINO – A New Petri Net Software Tool for Control of Discrete-event and Hybrid Systems Using Arduino Microcontrollers. In: *Proceedings of the Federated Conference on Computer Science and Information Systems*, Leipzig, Germany, 915–919.
Lee, E. A. (2017). *Plato and the Nerd: The Creative Partnership of Humans and Technology*. The MIT Press.
López-Fernández, D., Gordillo, A., Alarcón, P.P., Tovar, E. (2021). Comparing Traditional Teaching and Game-Based Learning Using Teacher-Authored Games on Computer Science Education. IEEE Transactions on Education, 64, 367–373.
Master J., Patterson E., Yousfi S., Canedo A. (2020). String Diagrams for Assembly Planning. In: Pietarinen AV., Chapman P., Bosveld-de Smet L., Giardino V., Corter J., Linker S. (eds) *Diagrammatic Representation and Inference. Diagrams 2020*. Lecture Notes in Computer Science, 12169, Springer, Cham.
McCabe, A., O'Connor, U. (2014). Student-centred learning: the role and responsibility of the lecturer. *Teaching in Higher Education*, 19(4), 350–359.
Moore, K. (2018). Minecraft Comes to Math Class. *Mathematics Teaching in the Middle School*, 23, 334, DOI: 10.5951/mathteacmiddscho.23.6.0334.
Mørch, A.I., Mifsud, L., Eie, S. (2019). Developing a Model of Collaborative Learning with Minecraft for Social Studies Classrooms Using Role-play Theory and Practice. In: *CSCL*, 17–19.
Murata, T. (1989). Petri nets: properties, analysis and applications, In: *Proc. of theIEEE*, 77, 541–580.
Panja, V., Berge, J. (2021). Minecraft Education Edition's Ability to Create an Effective and Engaging Learning Experience. *Journal of Student Research*, 10(2). DOI: 10.47611/jsrhs.v10i2.1697.
Plass, J. L., Homer, B. D., Kinzer, C. K. (2015). Foundations of game-based learning. *Educational Psychologist*, 50(4), 258–283.
Ponce Carrillo, R., Alarcón Pérez, L. M. (2020). Virtual environments for academic writing. A model in Minecraft. *ALTERIDAD. Revista de Educación*, 15(1), 76–87.
Prayaga, L., Davis, J., Whiteside, A., Riffle, A. (2016). An Exploration In The Use Of Minecraft To Teach Digital Logic To Secondary School Students. *International Journal of Computer Science*, 2(2).
Reis, A.C.B., Barbalho, S.C.M., Zanette, A.C.D. (2017). A bibliometric and classification study of Project-based Learning in Engineering Education. *Production*, 27(spe), e20162258.
Šajben, J., Klimová, N., Lovászová, G. (2020). Minecraft: Education Edition as a Game-Based Learning in Slovakia. In: *The Proceedings of 12th Annual International Conference on Education and New Learning Technologies*, 1–8, DOI: 10.21125/edulearn.2020.1946.
Schmitz, D., Moldt, D., Cabac, L., Mosteller, D., Haustermann, M. (2016). Utilizing Petri Nets for Teaching in Practical Courses on Collaborative Software Engineering. In: *Proceedings of the 16th International Conference on Application of Concurrency to System Design*, Torun, Poland, 74–83.
Siddiqui, N., Dave, R., Seliya, N. (2021). Continuous Authentication Using Mouse Movements, Machine Learning, and Minecraft. *ArXiv*, abs/2110.11080.
Silva, M. (2013). Half a century after Carl Adam Petri's Ph.D. thesis: A perspective on the field. *Annual Reviews in Control*, 37, 191–219, DOI: 10.1016/j.arcontrol.2013. 09.001.
Singh, S. (2020). Minecraft as a Platform for Project-Based Learning in AI. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 13504–13505, DOI: 10.1609/aaai.v34i09.7070.
Solomon, C.A., Harvey, B., Kahn, K., Lieberman, H., Miller, M.L., Minsky, M., Papert, A., Silverman, B. (2020). History of Logo. In: *Proceedings of the ACM on Programming Languages*, 4, 1–66.
Taub, M., Sawyer, R.K., Smith, A., Rowe, J.P., Azevedo, R., Lester, J.C. (2020). The agency effect: The impact of student agency on learning, emotions, and problem-solving behaviors in a game-based learning environment. *Comput. Educ.*, 147, 103781.
Thong, W.J., Ameedeen, M.A. (2015). A survey of Petri net tools. In: *Advanced Computer and Communication Engineering Technology*, Springer: Cham, Switzerland, 537–551.
Toquero, C.M. (2021). Emergency remote education experiment amid COVID-19 pandemic. *IJERI: International Journal of Educational Research and Innovation*, 15, 162–176, DOI: 10.46661/ijeri.5113.
Vörös, A., Darvas, D., Hajdu, Á., Klenik, A., Marussy, K., Molnár, V., Bartha, T., Majzik, I. (2018). Industrial applications of the PetriDotNet modelling and analysis tool. *Sci. Comput. Program*, 157, 17–40.
Walsh Jr., T. (2019). Exploring Computer Science with MicroworldsEX to Learn Geometry and Logo Programming Code. *Theory and Practice: An Interface or A Great Divide?*
Wisniewski, R., Grobelna, I., Karatkevich, A. (2020). Determinism in Cyber-Physical Systems Specified by Interpreted Petri Nets. *Sensors*, 20, 5565, DOI: 10.3390/s20195565.
Worsley, M., Bar-El, D. (2020). Spatial Reasoning in Minecraft: An Exploratory Study of In-Game Spatial Practices. *Computer Supported Collaborative Learning*, 2, Retrieved from https://par.nsf.gov/biblio/10170333.
Yu, Q., Cai, L., Tan, X. (2018). Airport Emergency Rescue Model Establishment and Performance Analysis Using Colored Petri Nets and CPN Tools. *Int. J. Aerosp. 2018*, 2858375.

I. Grobelna received the M.Sc. and Ph.D. degrees in computer science from the University of Zielona Góra, Poland, in 2007 and 2012, respectively. She is currently an Assistant Professor with the Institute of Automatic Control, Electronics and Electrical Engineering, University of Zielona Góra. She has authored three books and co-authored over 50 scientific papers. Her current research interests include design, specification and verification of automation control systems.

M. Mazurkiewicz received the M.Sc. and Ph.D. degrees in computer science in 1999 (Technical University of Zielona Góra, Poland) and 2007 (University of Zielona Góra, Poland), respectively. Since 2007, she has been an Assistant Professor at the Faculty of Computer, Electrical and Control Engineering of the University of Zielona Góra. Her general research interests include methods of digital circuit synthesis, design and PLC programming.

D. Janus is currently a student at the Faculty of Computer, Electrical and Control Engineering, University of Zielona Góra, Poland. His interest in playing games helped to incorporate the methods into Minecraft.