Exploring Different Unplugged Game-like Activities for Teaching Computational Thinking

Tomislav Jagušt, Ana Sovic Krzic,
Gordan Gledec, Mislav Grgić
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, Zagreb, Croatia
e-mail: {tomislav.jagust, ana.sovic.krzic, gordan.gledec,
mislav.grgic}@fer.hr

Iva Bojic
SENSEable City Laboratory,
Singapore-MIT Alliance for Research and Technology,
1 Create Way, Singapore, Singapore
e-mail: ivabojic@mit.edu

***Abstract* — For a number of years, various games have been used as an educational tool at different academic levels, especially in primary education. However, only recently games that teach coding and algorithmic thinking or even broader, computational thinking, emerged. Initiatives like Hour of Code and similar on-line activities or block-based programming games popularized the field, while at the same time, projects like CSUnplugged showed that the "idea of programming" can be learned even without using the computer. In this paper, we present our experience so far in creating and implementing different unplugged activities that teach students of different age levels, from early primary school to the university students, the simple programming concepts and algorithms. As a part of Science, Technology, Engineering, Mathematics and Computer Science (STEM-C) outreach program named SUZA - From school to science and the academic community at University of Zagreb Faculty of Electrical Engineering and Computing, Croatia, we conducted a number of game-like activities based on graph paper programming, block-based programming and using the existing board games. Although this is a work in progress project, the participant reactions so far, collected through interviews and questionnaires, indicated that the conducted activities were well accepted by students and their teachers. We also received a number of useful feedback comments and proposals, such as to expand the activities to include the homework part, or to make them more physical and relocate them outdoors.**

Keywords—STEM-C; computational thinking; algorithmic thinking; unplugged programming; education;

I. INTRODUCTION

For several years now, at the University of Zagreb Faculty of Electrical Engineering and Computing in Croatia, exists a science popularization and outreach program called SUZA - From school to science and the academic community [1]. As a part of the program, a number of various activities were organized, from Faculty Open Day through different science popularization lectures, lifelong learning sessions for Computer Science (CS) and technical education teachers, to school visits and guided tours around faculty laboratories [2]. Our volunteers come from various backgrounds - university professors, research stuff, PhD students, undergraduates, etc. [3]. An important part of activities, which are organized under the umbrella of this program, is participation in various science fairs and similar science popularization events.

These events very often are organized in open spaces, museums, libraries, school halls and similar locations. Typically, in such places, there is a lack of Information and Communication Technology (ICT) equipment or simply it is difficult to set up and supervise the usual CS laboratory or learning environment. Moreover, the type of the audience and time that the participants spend at each fair stand is rather short, so the presented activity has to be "appealing to the eye" and last for a maximum of a few minutes. Additionally, many primary and secondary schools in Croatia, especially outside of urban areas, are not adequately equipped, and CS classes are usually performed by splitting a normal class in two smaller groups, so each student (or two students working in a pair) could have her own computer.

For these reasons a few years ago, we decided to explore and adapt a number of existing unplugged activities which were designed to motivate and encourage students for different areas of CS, like programming and algorithmic and computational thinking. In this paper, we describe four different activities, inspired by existing lectures or children games and tailored for our specific scenarios.

II. RELATED WORK

Learning and games are two very closely related terms. Most of the early childhood learning is happening through games and play, the practice that often continues during the entire human life. Through games, children discover the world around them, develop different skills, improve imagination, and experience many emotions. That is why learning through play is considered to be one of the most effective ways of learning. Games are, by definition, entertaining and fun, and as such can create a positive reaction, even of less interesting or boring activities. This increases intrinsic motivation and activities that include games become more enjoyable and interesting [4]-[7].

The importance of games and play in the cognitive development of an individual was first noted by Piaget and Vygotsky [8], and over time, an interdisciplinary area of Game-Based Learning (GBL) emerged. GBL examines the relationship between learning, games and playing, and is often described as "game with defined learning outcomes" [9]. Most authors agree that learning CS through games or using tangible

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

objects can improve students' motivation and engagement, which in end results in better test scores [10], [11].

Computational and algorithmic thinking are fundamental skills in the CS education. Often used interchangeably, although computational thinking is a broader form, these two skills are becoming the most important abilities of a 21st century workers [12]. Coined in 2006 by Jeannette Wing from Carnegie Mellon University, Computational Thinking (CT) is described as "universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" [13]. Although the whole area of CT is rather young, most researchers agree that it should become a part of obligatory curriculum for students of all ages [12]. Algorithmic Thinking (AT), on the other hand, puts focus on CS, especially the computer programming. It is often described as a set of skills connected with creation and interpretation of algorithms, like problem analysis and specification, division of a bigger problem into a list of steps or smaller and simpler problems, etc. [14]. As such, AT represents a crucial part of CT, and should be present in every CT lecture or learning material.

In recent years, largely thanks to global initiatives like Hour of Code[1], European code Week[2] or Scratch day[3], CT and AT are presented to young learners as small and interesting computer games, often including characters from popular culture, like Angry birds, Minecraft or Frozen. Although the online educational games like these are very popular and interesting, there are some scenarios where this type of CS learning is not feasible or appropriate. Firstly, there are still many schools that lack adequate equipment, and secondly, at specific events (like Science fairs), real-life, tangible and group activities are preferred over single player computer games. This gap is filled with unplugged CS learning activities. Typically, these activities include pencils and papers, cups or other tangible objects, and can be played in pairs or within a group. One of the biggest collections of unplugged activities from the CS field is CSUnplugged[4], although there are many other resources with similar ideas and learning materials, notably Institute of Electrical and Electronics Engineers (IEEE) TryEngineering[5] and TryComputing[6] or Hour of Code CS lessons without the use of technology[7].

III. Case Studies

In this section, four different case studies and implementations of unplugged activities are described. The activities were inspired by existing online lectures or games and adapted to fit the specific use case.

A) Graph paper programming meet Pac-Man

Using graph paper to play different games has been a popular activity for years. Some of famous children games were designed or can easily be played this way, like Battleship, The dot game or even a well-known Tic-tac-toe. As a part of Hour of Code initiative, a number of unplugged lessons that teach CS was developed. Some of them, like Coding: Algorithms [15] and Graph Paper Programming [16] use simple pencil & paper activities to teach the basics of CS and AT. We combined and extended these lessons with additional activities in a form of small games, in which students have to follow the written program to draw an object in an empty graph paper matrix, write their own program that will guide Pac-Man through the maze to the strawberry, or find out which of the given programs is correct (Figure 1).

The lessons were implemented in a number of small workshops, in combination with similar paper programming activities or as a prequel to similar computer-based activities, like Scratch, Python or Lego Mindstorms programming.

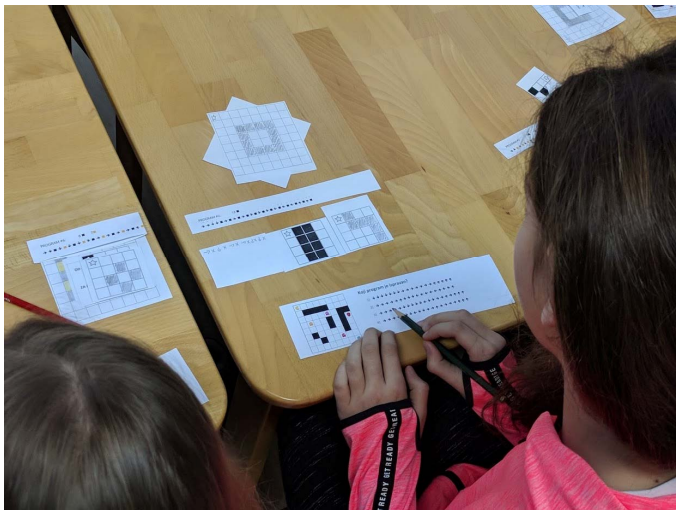

Figure 1 - student finding the correct program to guide Pac-Man through the maze

B) Life-size Graph paper games

For open-space events like Science fairs or Faculty open day a modified life-size version of Graph paper activity was implemented. The playing board was created as an 8x8 matrix, each field sized 30x30cm, created with a masking tape. A number of square papers (easily cut out from A3 paper sheets) of different color, together with printout Pac-man and Strawberry (figures) was prepared. In a single-player game, a player receives a number of square sheets and a piece of paper with a written program (programs from Case study A can be reused) and has to "execute" the program to draw a simple shape (e.g. a smiley, flower or heart) in the playing board. In a two player game, one player ("the programmer") reads statements of the program while the other player ("the robot") executes them (Figure 2). Another variant includes predefined obstacles on the floor and random position of Pac-Man and a strawberry. One player has to write the program for Pac-Man to reach the strawberry while the other player tests it by executing the written statements. Finally, another multiplayer variant incudes big sponge dice and a set of cards with simple statements: "step forward", "turn left", "turn right", "repeat statement until you hit the obstacle" and "execute the statement

[1] https://hourofcode.com/
[2] http://codeweek.eu/
[3] https://day.scratch.mit.edu/
[4] https://csunplugged.org/en/
[5] http://tryengineering.org/
[6] http://www.trycomputing.org/
[7] https://code.org/curriculum/unplugged

for N times". Players are split into two teams, and their figures together with random obstacles are placed in opposite corners of the playing board. Teams take turns throwing the dice, and the rolled number determines number of cards they receive in that turn to program their Pac-man. The team which reaches the strawberry first, wins (Figure 3).

Figure 2 - A group of students drawing the Smiley face shape on a Science fair

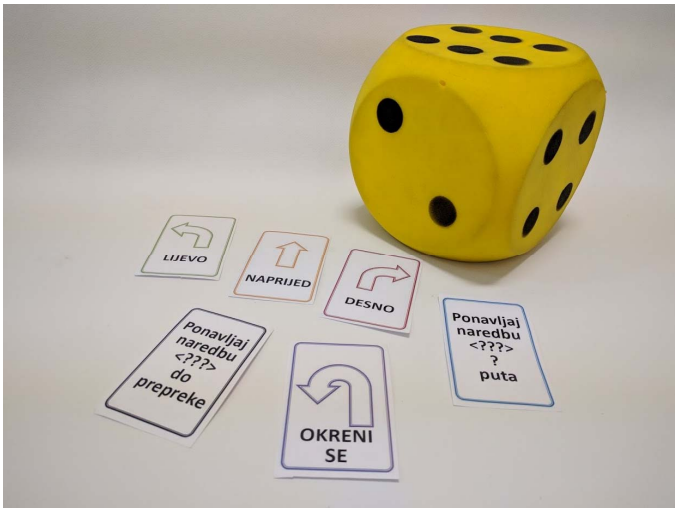

Figure 3 - Props for a real life graph programming game

C) The Network

Web pages www.tryengineering.org and trycomputing.org, maintained by IEEE, are great repository of learning lessons, and workshop plans. They provide answers to all sorts of questions and other resources for students of all ages, but also for teachers, school counselors and parents. One of the most popular lessons on the site is "Networks" [17] - a lesson that introduces students to the basics of computer networks, graph theory and Internet. The original lesson plan combines YouTube videos with computer exercises and unplugged activities.

For the purpose of a research project that included 8 primary schools and around 1100 pupils, we had to organize and implement a series of 45-minute interactive lessons, but without the use of computers. We decided to adapt the Networks activity, since it was well aligned with the curricula (computer networks are taught in the 6th grade). All computerized activities were removed, and instead a short introductory lecture using a whiteboard was prepared. The package transmission through the network (created on the floor with masking tape and post-it papers) was simulated with anti-stress balls, which also made it possible to illustrate a number of network problems, e.g. dropped packages (by throwing many balls at once to a student) or a broken link (by removing one of the masking tape lines) (Figure 4). Furthermore, the exercise was used to introduce the basics of graph theory and network routing to students, e.g. finding the shortest path between two nodes and assigning a cost to a specific edge/link.

The second part of the activity included a small graph programming task, similar to the one described in the case study A - students received a paper with 4x4 matrix and already drawn simple shape. Their task was to write the program that draws the given shape and then send the program through the network to other student in a group. The receiving student would then execute the program on empty 4x4 matrix and compare their result with the sender (Figure 5).

Figure 4 - Students exchanging the papers with graph paper programs over the "network"

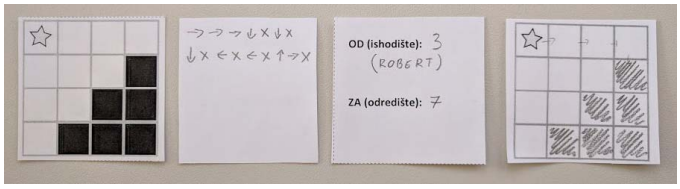

Figure 5 - From left to right: a simple graph programming task, the program, "network package header", and solution

D) Using board games to develop algorithmic thinking

Recently, a number of commercial board games that can be used to teach programming and algorithmic thinking emerged. After trying out different games with a group of 24 third year students at our faculty's "Science popularization" course, we

decided to offer one of the games (RoboRally) as a part of professional development training for 20 primary school ICT teachers (Figure 6).

In the game, players have to program small robots with a number of simple cards (similar to the game described in Case study B and Figure 3), avoiding a number of obstacles and interacting with surroundings. The rules were simplified to put more focus on programming, order of execution and causality, and less focus on other aspects of the game.

Teachers' reactions were positive so the workshop was repeated with another group of 20 teachers and there are plans to organize thematic programming board game events for teachers and students in future.

Figure 6 - Teachers playing the simplified RoboRally game

IV. Participants' Reactions

The overall participants' reactions to four presented activities were very positive. The games turned out to be very engaging, and real life activities attracted a lot of interest from science fairs visitors. Even the students without any prior experience in programming quickly mastered the simple loops and started using them in some non-trivial ways (e.g. to turn left by turning right three times).

Students particularly like Real life Pac-Man and The Networks and some of them planned to create a similar setup in a school yard using a colored chalk and plastic cups. As the main reason why they liked these activities, the students pointed out their tangibility which helped them to visualize the abstract terms like programming statements or computer network traffic more easily.

Three months after the teacher workshops held at our university that consisted of case studies C and D, we conducted a short online survey to see if the teachers implemented any of the workshops or approaches in their classes. Among 20 teachers that participated (which is about 25% of teachers that participated at the workshop), 11 of them said they tried at least one workshop and 9 respondents have not tried any of the workshops yet. A few selected comments are given below:

- The workshops were very useful to the teachers and to the students. Pupils were delighted, and very enthusiastic about the practical exercise. They wanted to repeat the exercise with a larger number of packages.
- It would be great if you can organize more such workshops that we could use in our lectures or provide more worksheets that we could give students as additional lesson material or as a homework.
- The workshops are excellent because they engage the students and make them to learn by their own experience. I noticed, after the Networks workshop with my students, that even after a year, none of the pupils has forgotten what packet transfer is and why network messages are shared using packets. Even those "bad" students...
- My 6th grade students have learned about networks in the way you showed us and it was very interesting to them. They understand everything much faster. I used examples of non-computer programming for motivation and as an introduction to programming in the 5th grade.
- I was particularly impressed by the networks. In fact, I did something like that before, but I just drew it on the board. Presented workshop seems better. Students' involvement is bigger.

Some of the reasons why teachers did not use the presented workshops with their students are:

- Lack of time (several similar answers).
- Incompatibility with planned lectures and program (several similar answers).
- Students like to do such activities, but despite all the effort they perceive it as something fun and unrelated to real life. So, often, the result of the activity is only distraction (waste of time) rather than learning.

V. Conclusion and Future Work

In this paper we presented our experience in implementing different unplugged activities that teach CS related topics, with focus on problems that develop computational and algorithmic thinking, from writing and executing a simple program, through practicing the usage of different loop types to the more sophisticated problems like finding a shortest path in a graph. The reception of activities was very good, both by students and parents, as well as by teachers. A number of teachers that participated in our workshops implemented some of the activities in their classes. As a future work, we plan to increase the number of different activities, as well as organize regular quarterly board game events.

Acknowledgment

This work has been fully supported by University of Zagreb Faculty of Electrical Engineering and Computing. We also acknowledge the support of the National Research Foundation, Prime Minister's Office, Singapore, under its Campus for Research Excellence and Technological Enterprise programme, Singapore-MIT Alliance for Research and Technology Future Urban Mobility Interdisciplinary Research Groups.

REFERENCES

[1] I. Bojic, T. Jagust, and A. Sovic, "Selected examples of cooperation between universities and schools in STEM education," *ISEC 2015 - 5th IEEE Integr. STEM Educ. Conf.*, pp. 189–194, 2015.

[2] I. Bojic and J. F. Arratia, "Teaching K-12 students STEM-C related topics through playing and conducting research," *Proc. - Front. Educ. Conf. FIE*, vol. 2014, 2015.

[3] T. Jagušt, A. S. Kržić, A. Nakić, M. Grgić, and I. Bojic, "What (de)motivates one to volunteer in K-12 STEM-C outreach activities?," *Proc. - Front. Educ. Conf. FIE*, vol. 2017–Octob, pp. 1–5, 2017.

[4] S. Deterding, R. Khaled, L. Nacke, and D. Dixon, "Gamification: toward a definition," *Chi 2011*, pp. 12–15, 2011.

[5] D. R. Flatla, C. Gutwin, L. E. Nacke, S. Bateman, and R. L. Mandryk, "Calibration games," *Proc. 24th Annu. ACM Symp. User interface Softw. Technol. - UIST '11*, no. June 2017, p. 403, 2011.

[6] M. S. Kuo and T. Y. Chuang, "How gamification motivates visits and engagement for online academic dissemination - An empirical study," *Comput. Human Behav.*, vol. 55, pp. 16–27, 2016.

[7] B. Monterrat, Élise Lavoué, and Sébastien George, "Motivation for Learning - Adaptive Gamification for Web-based Learning Environments," *Proc. 6th Int. Conf. Comput. Support. Educ.*, no. APRIL, pp. 117–125, 2014.

[8] A. Nicolopoulou, "Play, Cognitive Development, and the Social World: Piaget, Vygotsky, and Beyond," *Hum. Dev.*, vol. 36, no. 1, pp. 1–23, 1993.

[9] D. W. Shaffer, K. D. Squire, R. Halverson, and J. P. Gee, "Video Games and the Future of Learning," *Phi Delta Kappan*, vol. 87, no. 2, pp. 104–111, 2005.

[10] G. Kiss and Z. Arki, "The Influence of Game-based Programming Education on the Algorithmic Thinking," *Procedia - Soc. Behav. Sci.*, vol. 237, no. June 2016, pp. 613–617, 2017.

[11] G. Futschek and J. Moschitz, "Learning algorithmic thinking with tangible objects eases transition to computer programming," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7013 LNCS, pp. 155–164, 2011.

[12] J. Lockwood and A. Mooney, "Computational Thinking in Education: Where does it fit?," pp. 1–58, 2017.

[13] J. M. Wing, "Computational Thinking," vol. 49, no. 3, pp. 33–35, 2006.

[14] G. Futschek, "Algorithmic Thinking: The Key for Understanding Computer Science," pp. 159–168, 2006.

[15] Code.org, "Algorithms." [Online]. Available: https://studio.code.org/unplugged/unplug4.pdf.

[16] Code.org, "Graph Paper Programming." [Online]. Available: https://curriculum.code.org/csf-1718/coursed.pdf.

[17] IEEE TryEngineering, "Networks."