Immersively Learning Object Oriented Programming Concepts With sCool

Chanelle Kaith Mosquera
Computer Science Department
California Polytechnic State University
San Luis Obispo, CA, USA
cmosquer@calpoly.edu

Alexander Steinmaurer
Computer Science Department
Graz University of Technology
Graz, Austria, EU
a.steinmaurer@student.tugraz.at

Christian Eckhardt
Computer Science Department
California Polytechnic State University
San Luis Obispo, CA, USA
eckhardt@calpoly.edu

Christian Guetl
Computer Science Department
Graz University of Technology
Graz, Austria, EU
c.guetl@tugraz.at

***Abstract*—The proposed project, sCool, is an adaptive mobile game designed for STEM education. In this work, we present a new iteration of sCool in efforts to further examine contributing factors of engagement and comprehension. The new developed game experience for acquiring object-oriented programming skills is divided into two parts: concept-learning and practical. In the concept-learning part, the users explore a virtual environment filled with puzzle-pieces hinting theoretical concepts, preparing the users for the practical part. Users need to solve a programming challenge in Python, directing game-objects on a checkerboard to certain fields. Conducting a study involving 39 school students and three teachers, we are able to successfully display an enhanced understanding of different programming concepts.**

***Index Terms*—game-based learning, K-12 education, coding, digital skills**

I. Introduction

Digital skills become increasingly important in our modern society. For example, the European Commission published a digital competence framework for its citizens [1] and UNESCO a global framework for digital literacy skills [2]. This set of competences is seen as crucial for personal fulfilment and development, social inclusion and employment, and for active citizenship. Developing digital competences is even perceived as a human right. Competence areas include devices and software operation, information and data literacy, communication and collaboration, digital content creation, security, problem solving, and career-related applied competences.

Computational thinking and coding constitute important skills within the framework of digital competences. Various teaching and training approaches for formal education, lifelong learning and vocational training have been developed, researched and applied in educational settings [3]. Game-based learning in this context can provide fun, engaging and motivating but yet effective learning experiences.

The situation mentioned above has motivated us to start the research project sCool, which was originally initiated by research groups from Graz University of Technology in Austria and Westminster University in the United Kingdom [4]. Based on the exploratory learning model [5] and the MDA framework [6], sCool is designed as a mobile game to learn and practice coding in a playful way for school and novice bachelor students. Additionally, a web interface enables teachers and instruction designers to create their own learning content. Studies, both in school and for beginners at university level, revealed engagement and motivation and acquisition of knowledge of programming concepts and coding skills [7].

Based on promising previous results and experiences gained, new game types for teaching object-oriented programming skills could be added to the game. In addition this approach also implemented a more play-full way for acquiring text-based learning content. The main contributions in this research are:

- Introducing a new type of game for the mobile video game sCool
- Giving an overview of a game-based approach to learn object-orientated programming in game-based way
- Evaluating the new game types in three groups in order to get a comprehensive picture of different aspects (usability, engagement, usage in education, etc.)
- Receiving a pedagogical point of view for the developed concept from teachers
- Discussing findings to improve the current version for its practical usage in education

The remainder of this paper is structured as follows: Chapter 2 gives an overview of the relevance of computational thinking in education and introduced various approaches found in education. The previous work on the video game sCool version 2 is briefly described. Chapter 3 covers the requirements on the new game type of sCool version 3 and gives an overview of the further development. In chapter 4 the evaluation of sCool version 3 is presented in detail. This chapter includes

978-1-7348995-0-4/20/$31.00 ©2020 Immersive Learning Research Network
6th International Conference of the Immersive Learning Research Network (iLRN 2020)
Online, June 21-25, 2020.

an introduction of the used instruments, the procedure and describes the participants. The last section of this chapter deals with the results of the study. Chapter 5 summarize the findings and gives a prospect on future work.

II. Related Work

A. Computational Skill Teaching

Computational thinking is related to problem solving in a structured way. These techniques are general strategies to think about problems analytically. Computational thinking is not necessarily related to programming and computer science [8]. It is rather a higher-order way of think about problems [9].

There are different approaches in literature to define a set of skills related to computational thinking. A common classification is decomposition, pattern recognition, abstraction, and algorithms [10], [11]. These skills can be useful in other subjects as well and are not restricted to an academic point of view on computer science. Many national K-12 curricula already cover various approaches to teach student's in computational thinking. These national curricula distinguish in various aspects [12]. For one thing, there is no common term for computing education (CS, CSE, ICT, Informatics, Digital literacy, etc.), for another thing the objectives are different. General goals are digital literacy, algorithmic and logical thinking, problem solving, and understand basic computer science concepts.

Besides national efforts in creating a curriculum for computer science class different organizations work on standards as well. The Computer Science Teachers Association (CSTA) released the CSTA K-12 Computer Science Standards [13] in 2017 in order to provide a comprehensive curriculum for computer science. It is a standard for all students in K-12 education and is subdivided into five levels (1A, 1B, 2, 3A, and 3B) that are assigned to a corresponding age group from kindergarten until grade 12. Each level consists of the same overall concepts:

- **Computing Systems**: Hardware, Software, Devices and Troubleshooting
- **Networks and the Internet**: Network Organization, Cybersecurity,
- **Data and Analysis**: Collection, Visualization, Transformation, Storage
- **Algorithms and Programming**: Variables, Algorithms, Control, Modularity, Development
- **Impacts of Computing**: Culture, Ethics, Safety Law, Social Interactions

B. Tools for Teaching

A wide range of tools help to teach various aspects in computing. Some tools cover just certain aspects of computing, others are orientated on curricula (for example CSTA K-12 curriculum). Many tools follow a game-based approach. Games with a focus on coding can be separated into three groups [14]:

1) *coding* understand programming language and its syntax
2) *algorithmic thinking* learn related concepts not depending on a certain programming language
3) *creating games* create own games in a technical way

The web-based tools CodeCombat [15] and CodeMonkey [16] are games of the category *coding*. The setting of both games is similar since the tasks are based on a narrative. These tasks have to be solved using a programming language (JavaScript or Python). Therefore, the players have to control an avatar using commands and apply introduced concepts like algorithms, loops, conditions, objects, etc. The game LightBot [17] is a puzzle game for learning computational concepts. The game's focus is to teach *algorithmic thinking* by using command blocks without a certain programming language. New concepts are introduced consecutively so the level's complexity increase. In this way concepts like procedures, recursion, loops, and conditions can be learned. The popular programming language Scratch [18] makes it possible to *create games* in a simple way. It is a block-based language and supports the usage of different media (sound, graphics, etc.). Students can create games based on different blocks (control, data, events, etc.).

The mobile video game sCool [4] was initial developed in a cooperation between Graz University of Technology and Westminster University to support students in computer science education. The idea was to develop a flexible game environment that supports an exploratory way of learning. The game-based approach should encourage players to explore the game and learn new concepts. The modular architecture enables an agile way to add more game types and concepts based on existing content. The game has two different modes with a certain purpose. In the concept-learning (or explorative) game mode new concepts are introduced in a textual way. This exploratory approach should encourage players to learn new concepts that have to be applied in the practical mode, which is the second mode. Both game modes are linked via an overall narrative. The goal is to support a space shuttle's crew to repair the broken shuttle and escape from the foreign planet. Based on several evaluations [7] the practical missions support different coding concepts. The practical mode supports pseudo block-based programming where the blocks are converted into Python code. This makes the interaction with the code editor for beginners easier since they do not have to focus on syntactical details. Educators can define the concepts using a web application to have a highly adaptive content. Players can learn concepts like sequencing, conditions, loops, objects, data types, etc. Based on the experiences in school and evaluations with educators it was decided to extend the existing game with new game types. One of the central improvements was to change the way concepts are textually presented to the students because they often did not read them. Another requirement was to introduce object-oriented concepts in a meaningful way that students are able to understand and apply basic concepts in an intuitive way.

III. Computational Skill Teaching Using sCool

A. Requirements and Concepts

This expansion of sCool promotes a new game type for both the concept-learning mode and the practical mode. ”Game type” refers to the genre of the game and its gameplay experience. In the concept-learning mode, the main task is to scavenge for resources, collect puzzle pieces which contain fragments of text, and solve the puzzle by stringing the text together in the correct order. In the practical mode, the player applies the concepts they learned in object-oriented programming to defeat enemies. The addition of these features propose a solution for improving player engagement in text-based learning and providing a structure that fosters learning of object-oriented programming concepts.

As mentioned earlier, sCool is built with a flexible architecture. This design enabled the thorough integration of this new game types. The game types can be played with existing players and lesson plans, in addition to new ones. It continues to use the existing web API to obtain content and then send the results back. It also maintains the in-game currency which was introduced as coins in the previous versions of sCool and comes back in this game type as resources. Therefore, any existing players can continue to earn currency playing this game types and spend it in the in-game store, a previous successful feature.

sCool’s flexibility is further demonstrated by its use of additional existing parameters set in the web platform. These parameters influence the game in the following ways:

- **Concept-learning part**: size and complexity of map, number of interactive objects
- **Practical part**: length of challenge, number of enemies, number of programmable units

The structure of the game design is based on the MDA framework and influenced by the Exploratory Learning Model, guiding the design of an immersive learning experience in a virtual environment. Experiential learning cycle is made of the following elements: exploring, reflecting, forming abstract concepts, and testing in new situations.

The player has the freedom to explore the environment, be introduced to theoretical concepts, apply them in a practical manner, then proceed to additional levels where they can further attain and practice their skills.

B. Development Details

This section goes into detail of the development of the concept-learning and practical parts of the game. In the game, a narrative is used to immerse players in the environment. At the game’s menu, the player can select which game type they prefer to play. Data integrity is maintained between each game type. The option to choose allows the game to appeal to different types of players and ways of learning. However, this paper focuses on the new game types which specifically appeals to learning object-oriented programming concepts.

The main goal of the concept learning mode is first to immerse and engage the players in a fun and interactive virtual world. Then, the player is taught some concepts from the lesson plan. This structure follows the exploration part of the exploratory learning model [5].

(a) Chopping the tree. (b) Collecting resources.

(c) Finding mystery chests.

Fig. 1. Concept-Learning Part - Exploring.

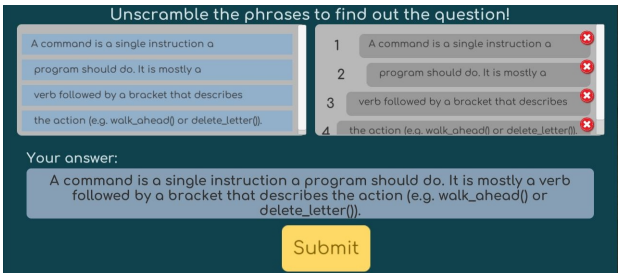

Fig. 2. Solving the Puzzle.

In the gameplay’s narrative, the player is a scavenger collecting resources outdoors. These resources are made into boxes which can be programmed to help the player protect the outdoors from enemies. To illustrate the narrative, the following describes the players’ experience in-game.

In the concept-learning part, the player finds themselves in a 3D outdoor setting where they explore the map and collect resources. Hidden among the resources are also mysterious chests the player needs to collect in order to complete the level. Fig. 1 demonstrates the sequence of events that follow the player’s tasks - chopping the tree, collecting resources, and finding mystery chests. The player interacts with the tree by pressing a button, resulting in collectable wood scattered across the ground and a mystery chest. When the mystery chest is collected, it reveals to contain a parchment of paper.

After all parchments of paper are collected, the player tries to piece the broken text together to make a cohesive text. Unscrambling the text reveals a concept for the students to learn and follows a multiple-choice question they must answer about it. If the player answers the question correctly, they move on to the next level; otherwise, they must repeat this level. As Fig. 2 shows, the collected text fragments are listed in scrambled order on the left, which the player must drag and drop onto the right side in the correct order. In order to unscramble the text, the player must read the fragment of text and choose the sequential fragment that will create a sensible sentence. This task requires attentiveness and reading

comprehension, motivating the students to actually read and comprehend the text, as opposed to simply skimming and moving forward with the game.

In the practical part of the game, the player must program the blocks to catch the enemies. At the challenge's initial state, there are wooden boxes positioned arbitrarily on an XY-coordinate playing field, as shown in Fig. 3. As time passes, bugs are popping up on the field and the player has a set time to catch them all. The total number of bugs that appear and the amount of time depends on the level's difficulty. The player must solve this challenge by programming the boxes to move to different points on the field where the bugs pop up. Since there are multiple boxes to interact with, the player must specify which instance of the box to move by referencing that specific box. This ability demonstrates this version of sCool's pseudo block-based programming adaptation to support object-oriented programming.

As Fig. 4 shows, the player is provided with a block-based programming tool where they must drag and drop the code block (left) onto the text editor (right). For this challenge, the available code blocks the player can use contains pseudo code for changing a block's label and for moving it. They then replace the pseudo code with correct Python expressions.

The system's support for object-oriented programming was developed by creating a class (which represent the boxes in the game) in C# and wrapping it with a Python class. The C# class allows access to the Unity libraries which are necessary for displaying and interacting with the boxes in the game, whereas the Python class allows interaction with this C# class using Python code.

Each box is an instance of this class and contains a string variable "Label" and a function "Move(int x, int y)". At the start of the game, the boxes are given default labels "block0" and "block1". The code in Fig. 4, shows that the player can access block0's label variable by calling "block0.Label" and setting it to a new string "i_23". In the proceeding lines, block0 is now referenced as i_23. Furthermore, to access a specific box object's MoveTo(...) function, the player must call the function from a specific block, such as in the second line "i_23.MoveTo(2,4)" and in fourth line "block1.MoveTo(8,8)".

This method of gameplay demonstrates to the students that boxes share similar properties, such as its label and ability to move. However, the values of each box's properties can differ. Therefore, boxes are simply instance of a Box object and are treated as unique entities with similar characteristics. This concept and way of thinking is explained through instructional text at the start of the level, and then put into practice in the gameplay. As a first-experience introduction to OOP, this perspective of seeing the objects as unique instances of the same type of object is beneficial for comprehending OOP concepts moving forward.

Fig. 5 shows the game's state after compiling and running the python code. The boxes in the scene have been relabeled accordingly and are moving to the desired destinations.

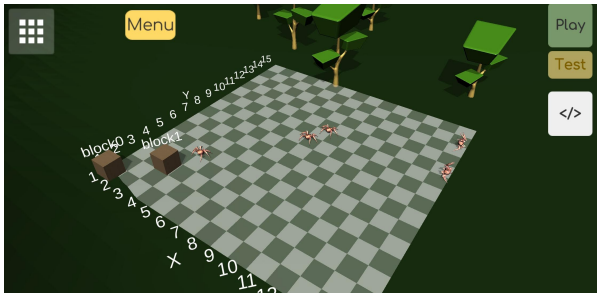

Fig. 3. Initial state of challenge.

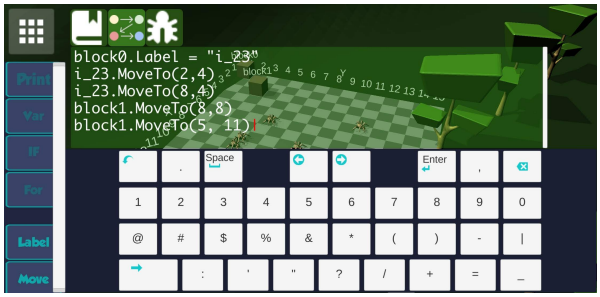

Fig. 4. Block-Based Object Oriented Programming.

IV. Evaluation in School

The focus of the initial study was to evaluate different aspects of the prototype and figure out if the new developed game types are suitable for computer science education. Therefore, we conducted a study on three different groups in two schools. The first school was a regular secondary school and the second one an academic secondary school (lower cycle). The student's age range is similar in both schools (7th grade in secondary school and 8th grade in academic secondary school). The choice of different school types should bring a variety in social background and experience in game-based learning approaches. There were four research questions related to the evaluation:

1) How are the new game types perceived by the students from different school types?
2) Is the user interface intuitive for students?
3) Can sCool's new game types help the students to understand certain coding concepts?

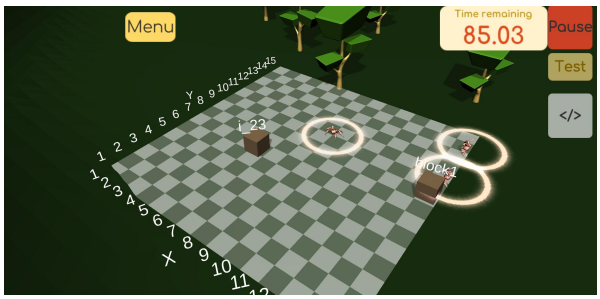

Fig. 5. After code execution.

TABLE I
CORRELATION OF CONCEPTS AND MISSIONS

Mission	Concept	Question
Concept-Learning Missions		
Task 1	Introduction	What are bugs?
Task 2	Objects	Which two terms characterize objects?
Task 3	Properties	What means label in terms of objects?
Task 4	Properties	Which command renames an object?
Task 5	Methods	Which command will move an object?
Practical Missions		
Task 1	Objects	Defeat five bugs using both blocks
Task 2	Objects	Defeat fifteen bugs using both blocks

4) What are educators' opinions about the new game types?

A. Setting and Instruments

This study was structured to be conducted in a classroom setting, consisting of different social forms. All interventions were in a supervised environment where one external supervisor and the class' computer science teacher were present the whole time. Each sequence was clearly separated from each other in order that all students know exactly what to do. In general the students were instructed to work in groups of two in order to help each other. Depending on the schools ICT infrastructure they worked together on one computer or each student on a separate one. sCool version 3 (which supports the game types introduced in this paper) was installed together.

The content was created using the sCool web application. In this way all students were provided with the same content. The course consisted of five concept-learning and two practical missions. Table I shows what concepts were linked to the corresponding missions. Overall the focus was on object-orientated programming using the Python programming language. The tasks in the concept-learning missions presented a new concept in a textual way. The text was separated in four chunks and the students had to unscramble the the text into one coherent answer. After completing this task a multiple-choice question based on the previous read text had to be answered. The asked questions should review if the students read the text explaining a concept. After the concepts were acquired, two practical tasks had to be solved. These missions required to do some programming on two block objects in order to defeat the bugs. The number of enemies was increased to increment the level of difficulty. The overall goal of all missions was to introduce students into object-oriented programming. Since the study was limited to 50 minutes, the participants should gain a basic understanding for objects and its properties and methods.

All data was collected based on the participants' consent. The data processing happened fully anonymous since the students worked with pseudonyms.

After working with sCool the students had to solve a similar task on a worksheet. This provides information about the level of knowledge and also shows if the students are able to understand the idea of objects and can apply them in a slightly changed problem on a different medium.

For the student's evaluation at the end of the experiment a comprehensive questionnaire was used. Each experiment included the same asked questions and they covered the following categories:

- general questions
- game engagement [19]
- system usability [20]
- game-related questions

The questionnaire was answered using Google Forms [21]. The data collected from general and game-related questions were analyzed using Microsoft Excel. Both game engagement and system usability was analyzed using the open-source programming language R to categorize the factors and calculate mean and standard deviation. The game-related data was sent from all computers to the server. All data is stored in a Microsoft SQL database. The data visualization was processed using sCool's web application and Microsoft Excel.

The teacher's evaluation was conducted after each experiment. For this purpose various questions were prepared beforehand and asked as part of a structured face-to-face interview. The focus of the interviews was to get a teachers opinion related to pedagogical aspects. At the end of the interview the educators could add any additional feedback. The information was collected and analyzed using Microsoft Excel.

B. Procedure

The three conducted experiments had the same structure. Table II illustrates the general schedule of all experiments. Each experiment was done in 100 minutes in total. After a short introduction and brainstorming about technology and coding in everyday life the game was introduced to the group. The students played the game for 50 minutes and were instructed to help each other if questions appear. Within the given time the students were presented five concept-learning and two practical tasks. After 50 minutes the students stopped playing the game and started with the worksheet. Fig. 6 shows the worksheet's task where the students had to navigate the bikes to the corresponding houses using already learned Python commands. The XY-coordinates should enable a setting the students are familiar with. Conclusively the students filled out the questionnaire.

C. Participants

The experiment was conducted in three groups of two schools. Table III gives an overview of the groups. The first group was a regular secondary school in 7th grade. In total twelve students (2 girls and 10 boys) attended the evaluation. They did not have any prior knowledge in programming and never worked with game-based approaches in computer science class. The second and third group were students of an academic secondary school in 8th grade. The groups consisted of 13 students (13 boys) and 14 students (6 girls and 8 boys). Three of the students in the second group had prior knowledge in programming (Scratch and HTML) and eight students in

TABLE II
EXPERIMENT PLANNING

Time	Task	Method
5’	welcome, introduction of the project	frontal
10’	motivation for programming, brainstorming	in plenum
5’	introduce sCool and game types	in plenum
5’	form groups, installation	group
50’	play sCool	group
10’	work on worksheet	individual
10’	answer questionnaire	individual
5’	final discussion	in plenum

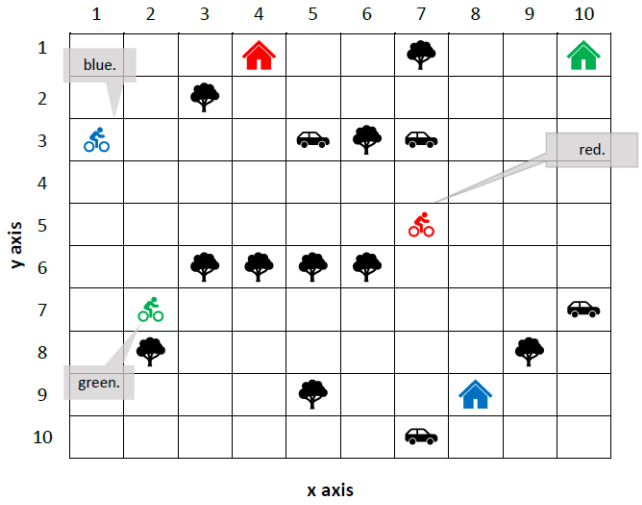

Fig. 6. Practical task on the worksheet after working with sCool.

the third group already programmed with Scratch. In total all groups covered 39 participants between 12 and 15 years (M=13.09, SD=0.61) consisting of 8 girls (20.5%) and 31 boys (79.5%).

To get a pedagogical point of view on the video game an evaluation with three teachers was also part of the study. Therefore, interviews with three different teachers were conducted. All participants differ in various aspects: gender, professional experience, education, and teaching methods. Table IV gives an overview of the interviewed teachers. Their professional experience is between 1 and 30 years. Each participant had a different education in teaching and computer

TABLE III
OVERVIEW OF EXPERIMENTS’ PARTICIPANTS

	Group 1	Group 2	Group 3
Grade	7th	8th	8th
Age	12–13 years (M=12.27) (SD=0.22)	12–15 years (M=13.69) (SD=0.39)	13–14 years (M=13.5) (SD= 0.27)
Gender	10 boys, 2 girls	13 boys	6 girls, 8 boys
Coding Experience	0 persons	3 persons	8 persons

TABLE IV
OVERVIEW OF INTERVIEWED TEACHERS

	Teacher 1	Teacher 2	Teacher 3
Gender	female	male	male
Experience	15 years	1 year	30 years
Education	college of education	university	IT certifications
School Type	secondary	secondary	secondary/college

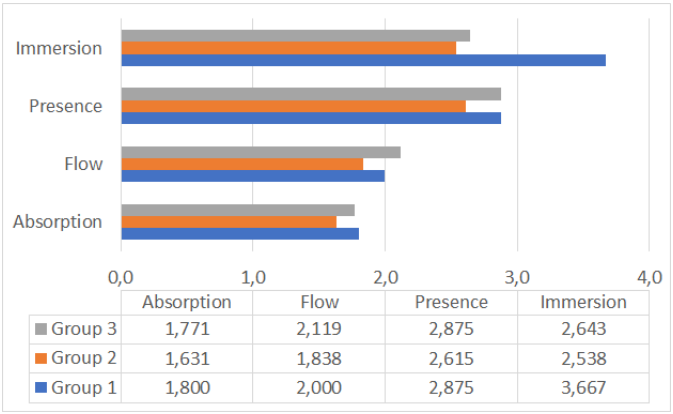

Fig. 7. Comparison of Game Engagement [19].

science.

D. Results and Discussion

In total 39 students (8 girls and 31 boys) attended the initial study. The general conditions for all groups were equal. When considering the results of the Game Engagement Questionnaire in Fig. 7 the distance between the level of immersion and presence is higher at the first experiment. This could be traced back to the fact that this students are not familiar with game-based approaches in computer science class and are more engaged in virtual environments. The level of absorption is similar in all groups.

The system usability score reached a value around (and greater) 68 in each group which means the usability is above average:

- SUS Group 1: 70 (SD=13.7)
- SUS Group 2: 67.9 (SD=14.6)
- SUS Group 3: 74.5 (SD=17.3)

All of the students play (mobile) video games and at least 30.8% even did programming in a game-based way before. Consequently they already used different systems so the System Usability Score can be taken as a good indicator for the students perspective on the system’s usability. The students were asked to rate the game’s control and the keyboard at the game-related questions. 87.18% agreed that the control is easy to use and even 92.31% answered that the usage of the keyboard is also easy. Some of the students asked for additional features in the code editor (mouse cursor, more keyboard support). The SUS score of each group and the students feedback regarding their experience approve that the user interface in general is intuitive for students.

31 students (79.49%) answered that programming with sCool was fun for them and 33 persons (84.62%) stated that they enjoyed this kind of game. All participating students were able to solve the concept-learning tasks of the game. 35 out of 36 players (groups) could pass the first practical task and another 28 groups (77.78%) could pass the second task as well. The most successful group was group 3 where all students could master all missions. The worksheet was assessed based on a correct syntax and each individual delivery task. 38 students solved at least one task correctly and 26 (66.67%) of these students even solved all tasks. The students had a high confidence in their self-assessment since 30 students (76.92%) answered that they learned something while playing the game. At least more than half (56.41%) of the participants stated that they got encouraged to learn more about programming.

All teachers are using apps in their class and they stated the following three aspects as most important:

- Add and create own content into the app
- Integrability in class
- Simple usage (availability, installation, etc.)

Although just one teacher does programming in class but each participant would use sCool as a tool for introducing computational skills and coding. All teachers answered that sCool would fit best to work on a new topic and learn to apply concepts. The teachers also agree that the game would be most appropriate for secondary school (especially grade 5 to 8). When asking for strengths and weaknesses the teachers answered that sCool is a good approach to motivate students to learn coding in a different way. Another advantage is that the teachers can focus on a student-centred class where the children can be coached. They see the disadvantages in the current design and user interface. A more intuitive way to make the game more self-explaining was requested. It was also considered that students need some interest in coding to make a successful learning experience.

V. Conclusion and Future Work

The goal of the continued design and development of sCool is to explore the educational impact of new game scenarios. In this iteration, the focus was on increasing the reading comprehension for text-based learning and to create a structure that supports the study of object-oriented programming. Conducting the Game Engagement Questionnaire and a post survey regarding the individual experience with middle school students as well as their teachers displayed a great acceptance of usability, high engagement, and an increased affinity to further investigate programming skills.

Based on our results, future work encompasses broadening sCool's support for more complex programming concepts and its usability for different learning environments. At its current state, sCool is played in conjunction with teacher instruction and additional assignments; however, a more self-explanatory game-play experience will allow sCool to be an option for those learning outside of a classroom setting. A long-term evaluation with two groups can help to get a comprehensive image over the game's fully educational potential. By using two groups the learning achievements of a group using sCool and a group with traditional coding class can be compared.

Acknowledgment

We want to gratefully acknowledge the support of Graz University of Technology and Cal Poly University for this research, but in particular the Marshall Plan scholarship supporting on the students involved in this research. We also want to thank Johanna Pirker, Milos and Aleksandar Kojic from Graz University of Technology and Markos Mentzelopoulou and Daphne Economou for co-initiating this research project

References

[1] S. Carretero, R. Vuorikari, and Y. Punie, *DigComp 2.1: The digital competence framework for citizens with eight proficiency levels and examples of use*, ser. EUR, Scientific and technical research series. Luxembourg: Publications Office, 2017, vol. 28558.

[2] N. Law, D. Woo, J. de la Torre, and J. Won, "A global framework of reference on digital literacy skills for indicator 4.4.2," UNESCO-UIS, Canada, Information Paper No. 51, Jun 2018.

[3] M. Romero, A. Lepage, and B. Lille, "Computational thinking development through creative programming in higher education," *International Journal of Educational Technology in Higher Education*, vol. 14, no. 1, p. 42, 2017. [Online]. Available: https://doi.org/10.1186/s41239-017-0080-z

[4] A. Kojic, M. Kojic, J. Pirker, C. Gütl, M. Mentzelopoulos, and D. Economou, "scool - a mobile flexible learning environment," in *iLRN 2018 Montana*. Verlag der Technischen Universität Graz, 2018, pp. 72–84.

[5] S. De Freitas and T. Neumann, "The use of 'exploratory learning' for supporting immersive learning in virtual environments," *Computers & Education*, vol. 52, pp. 343–352, 2008.

[6] R. Hunicke, M. Leblanc, and R. Zubek, "Mda: A formal approach to game design and game research," in *In Proceedings of the Challenges in Games AI Workshop, Nineteenth National Conference of Artificial Intelligence*. Press, 2004, pp. 1–5.

[7] A. Steinmaurer, "Revising a game-based learning platform for computational skills in education," Master's thesis, Graz University of Technology, Graz, Dec 2019.

[8] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, p. 33–35, Mar. 2006. [Online]. Available: https://doi.org/10.1145/1118178.1118215

[9] J. Mueller, D. Beckett, E. Hennessey, e. P. J. Shodiev, Hasan", and C. B. Hodges, *Assessing Computational Thinking Across the Curriculum*. Cham: Springer International Publishing, 2017, pp. 251–267.

[10] BBC. (2020, Jan.) Introduction to computational thinking. [Online]. Available: https://www.bbc.com/bitesize/eguides/zp92mp3/revision/1

[11] Code.org. (2020, Jan.) Cs fundamentals unplugged. [Online]. Available: https://code.org/curriculum/unplugged

[12] P. Hubwieser, M. N. Giannakos, M. Berges, T. Brinda, I. Diethelm, J. Magenheim, Y. Pal, J. Jackova, and E. Jasute, "A global snapshot of computer science education in k-12 schools," in *Proceedings of the 2015 ITiCSE on Working Group Reports*, ser. ITICSE-WGR '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 65–83. [Online]. Available: https://doi.org/10.1145/2858796.2858799

[13] Computer Science Teachers Association. (2017) CSTA K-12 Computer Science Standards, Revised 2017. [Online]. Available: http://www.csteachers.org/standards

[14] S. Combéfis, G. Beresneviuius, and V. Dagiene, "Learning programming through games and contests: Overview, characterisation and discussion," in *Olympiads in Informatics*, vol. 10. Vilnius University Institute of Mathematics and Informatics, 2016, pp. 39–60.

[15] CodeCombat Inc. (2019) Codecombat. [Online]. Available: https://codecombat.com/

[16] CodeMonkey Studios Inc. (2020) Codemonkey. [Online]. Available: https://www.codemonkey.com/

[17] LightBot Inc. (2017) lightbot. [Online]. Available: https://lightbot.com/

[18] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and et al., "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, p. 60–67, Nov. 2009. [Online]. Available: https://doi.org/10.1145/1592761.1592779

[19] J. H. Brockmyer, C. M. Fox, K. A. Curtiss, E. McBroom, K. M. Burkhart, and J. N. Pidruzny, "The development of the game engagement questionnaire: A measure of engagement in video game-playing," *Journal of Experimental Social Psychology*, vol. 45, no. 4, pp. 624 – 634, 2009.

[20] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[21] Google LLC. (2020) Google forms. [Online]. Available: https://www.google.com/intl/en_EN/forms/about/