ARCode: Augmented Reality Application for Learning Elementary Computer Programming

Sirawit Sittiyuno and Kornchawal Chaipah
Department of Computer Engineering
Faculty of Engineering, Khon Kaen University
Khon Kaen, Thailand

Abstract— **This paper presented ARCode, the system that employed augmented reality (AR) to motivate and help learners with programming. Since programming is an abstract task, many need help with programming, both technically and emotionally. ARCode is the game-based learning system that shows how each command works by using AR animations, and focuses on the logical order of commands. Learners can collaborate in the real world while practicing individually in the application. In our experiment, we found that the treatment group had significantly improved on their scores for all topics except for Selection, while the control group did not. The user satisfaction survey suggested that more than 80% of users accepted our system as a useful, enjoyable, and collaborative learning system. The primary results showed the system's potential to help learners better learn programming with high motivation.**

Keywords-Learning Programming, Augmented Reality, Mobile Learning, Game-based Learning.

I. INTRODUCTION

At present, many kinds of technology are controlled by computer software. We need personnel with high programming skills to reliably and effectively develop, adopt, and control these technologies. Not only software developers need programming skills, but so do other people. Now that computers have become parts of our lives, there have been movements to get everyone to program, or at least understand computer codes [1]. In order to take full advantages from computer technology, ones need to at least understand how it works. Moreover, programming involves systematic and logical thinking and design processes. These skills can help everyone living more effectively in everyday lives.

In general, we perceive a programming process as a problem solving process by employing computer languages as tools. A programmer needs to analyze a problem before implementing a program to achieve the desired results. However, our previous study [2] found that novice programmers had difficulties on the logical ordering of commands and systematically thinking, but not much on understanding the syntax. For some beginners, being unable to understand or visualize the workflow of the commands in a program makes it impossible to write the program correctly.

There have been many efforts to help beginners develop programming skills. There were applications simulating the actions of commands and the program workflow [3-6, 11]. However, these tools required the correct syntax before showing graphical simulations. So, they may not be suitable for those who do not have strong basic knowledge of syntax. Beginners could spend an ample amount of time debugging syntax instead of focusing on learning the logical flows of commands. Other works [1, 7] designed tools to use pre-defined block commands to control an animated character to complete a game-like goal. These works, such as in Code.org [1], allowed a user to instantly and clearly associate a behavior of a character to a specific command; thus, practicing thinking processes and enjoying the animation at the same time. Although these tools can engage a user to keep learning, most tools aim for simple goals and very basic commands. They might not help learners who want serious programming skills in the future by comprehending an actual computer language.

Therefore, in this work, we proposed a game-based AR mobile application aiming to help beginners to understand the workflow of a program, and provide a collaborative learning environment. The application employs Augmented Reality (AR) to visualize how programming commands work and shows the workflow of the program in order. This guides learners in the systematically thinking process. Del Bosque, Martinez, and Torres [9] and Teng and Chen [10] suggested that AR could help increase programming skills. In our case, we expected that AR would help make programming less abstract and more tangible by illustrating an analogy animation of a command behavior. For example, allocating a memory for a variable is analogous to creating a box to store a value. This new view of learning programming is expected to engage students by making students enjoy lessons, not overwhelming them with a page full of codes. However, because beginners need to get more advanced with coding in the future, the system also shows a command on an AR marker. This allows learners to associate the command with the corresponding animation. Moreover, learners can learn from each other in the real world, while working at their own pace in their mobile devices. This is expected to create a space for

978-1-7281-0719-6/19/$31.00 ©2019 IEEE

knowledge exchange, which is a suitable learning environment to increase programming skills [10].

In this work, we developed the system for three essential topics of C++: Fundamentals of C++, Selection, and Iteration. Our research questions are 1) whether the system help beginners improve programming skills by showing the operations of commands as AR animations, 2) whether the system help increase enjoyment and engagement in learning programming, and 3) whether can the system environment help student to learn together and improve cognitive and communication skills.

II. RELATED WORK

A. Augmented Reality Technology

Augmented reality (AR) is a technology that combines a real-world environment with a virtual world by creating graphics, such as images, videos, 3D objects, and text, overlaying on the real-world environment. AR is adopted in various fields, such as medicine, entertainment, and education. In education, AR aids learners to understand materials with media that transform abstract or intangible concepts to tangible ones. It also provides enjoyment and encourages users to learn materials. AR requires unique information to trigger virtual object presentation: marker (image), location (global coordinates) and panoramic 360. In our research, we use an AR marker-based type because it requires only markers, a camera, an AR engine and a screen, all of which can be contained in a smartphone, which is the targeted device for our learning system. Fig. 1 shows the working principle of an image-based AR. A camera first captures an image. A processor, an AR engine, then processes the image and maps the image with a pre-determined virtual object. Finally, an AR engine displays the virtual object by overlaying it on the real-world environment on the screen.

B. Tools for Learning Programming

There have been many tools helping beginners learn programming with various purposes, presentations, and supported languages. We classified these tools into code visualization, game and animation, and other tools. Code visualization tools usually simulate the execution of codes via graphics or animations. A user writes codes and then see the step-by-step simulations of what the codes are doing. The user can hence easily inspect if there is any logical error. Examples of these tools include Jeliot 3 [3], C++ Tutor [4], Code2flow [5], code visualization [6], and Java Program Flow Visualizer [12].

The game and animation-based tools usually required a user to place pre-defined commands or blocks of commands (block programming) to control animation characters. These tools allow users, especially young programmers, to immediately associate the effects of the commands with the actions of characters. Users can then easily determine the next steps of programming according to the previous actions of characters. Game-based learning satisfies users because they can learn with enjoyment, keeping them trying gradually more challenging lessons. It also encourages them to be more interested in lessons [12]. Examples of these tools are Code.org [1], Scratch [7], Program Your Robot [8], the games by Kazimoglu, Kiernan, Bacon, and Mackinnon [13], Fessakis, Gouli, and Mavroudi [14], and Gunter, Kenny, and Vick [15].

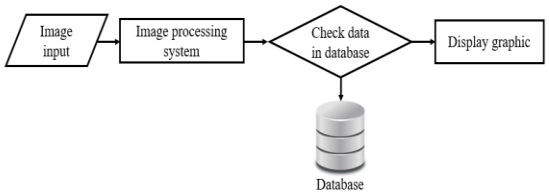


Figure 1. Working principle of an image – based AR

Other types of helper tools includes tools that help train programming skills by fixing programming bugs, adding missing codes to complete the program [16], using maps and flowcharts [17], puzzles [18], and using AR. Teng and Chen [10] developed an AR-based learning system to help students understand OpenGL programming. Learners can observe the rendered results, OpenGL codes, and the corresponding 3D objects in the virtual space by using QR markers. They could associate the relationships between OpenGL commands and resulting 3D object behaviors.

In this paper, we designed an AR game application to help learners with introductory C++ programming. The system used QR code markers, however, coupled with an actual command and a hinting analogy image of what the command is doing. We employed all advantages of the previous works of AR, block programming, code visualization, and gaming, to engage and help learners to improve their programming skills.

III. SYSTEM DESIGN

A. System Overview

We designed an AR game mobile application system, called ARCode, that helps C++ learners understand what an individual command does, such that they can logically order commands to correctly solve a problem. In our system, we provide a programming task and a set of paper markers. Each marker in the set represents one command. There are always more markers than needed. A user needs to scan the markers in a correct order to make a complete program. As Fig. 2 shows, after a user scans a marker using a mobile device, a corresponding animation clip is matched by an AR processing engine. The clip is then displayed on a user's screen overlaying the real world, demonstrating what the command does. A user can decide whether to save that command in the current program. The system allows a user to review all clips of the codes in sequence before submitting the codes. A user may edit the order of the commands in the command list before submitting the codes. The

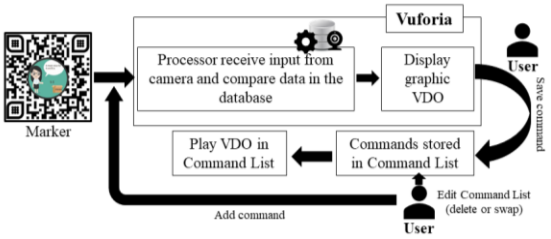


Figure 2. System architecture overview

system will then check whether the order of code is correct and give rewards and feedbacks to a user.

In addition, ARCode has features to challenge users to keep users trying to improve their programming skills. A user can review the play history on different topics, and challenge oneself by collecting medals for completing certain milestones.

B. System Design and Implementation

There are two main components in the system: paper markers and ARCode mobile application. As Fig. 3 shows, each marker contains a C++ command and QR code patterns coupling with an image. QR code patterns with an image provide enough details to differentiate markers, while an image gives an instant hint for a user of what the command does. Since the markers are paper-based, a user can download image files in the application and print them out for use.

In ARCode application, there are four main parts: AR platform, command list, code checker, and history and rewards. We developed the application to support Android OS version 4.4 (KitKat) and later. Fig. 4 shows the user interface and navigation design of the system. After a user logs in, the user can scan a marker and explore AR via Scan function, while the user can examine and edit saved commands via VDO list. The user can also go to History function to see previous plays and replay selected problems. Lastly, a user can see rewards via Reward function.

We employed Vuforia AR software platform [19] as an AR engine. After capturing a marker, an AR engine will process the marker image and find the matching clip in the database. Then the clip will be played and overlaid on a real world scene as in Fig. 5. Our AR animation helps a user understand how a command works by showing the command's behavior, right at the scanning moment. For example, the command `int cm;` means allocating a memory to store the value of a variable `cm;` similar to creating a box labelling `cm` to store the value of `cm`. AR is more interesting than the traditional all-code exercise, helping increase motivations in learning as well.

Having examined all marker choices for a given problem, a user scans selective markers and saves the corresponding commands into the command list, in the order as if the user is writing codes in a software editor. The command list helps train a user's systematic and logical thinking skills by allowing the user to add, delete, or swap commands, and see the effects of different orders of commands. As Fig. 6 a) demonstrates, the command list stores commands that a user saved previously from top to bottom. There is a video review section that the user can play videos to see how the codes are working sequentially in the current order, and examine whether the program would solve the given problem.

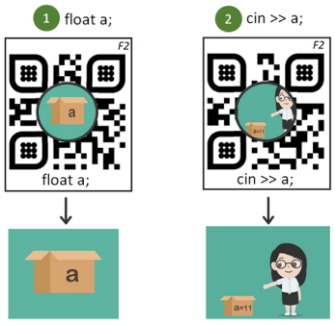


Figure 3. Mapping between a marker and a video

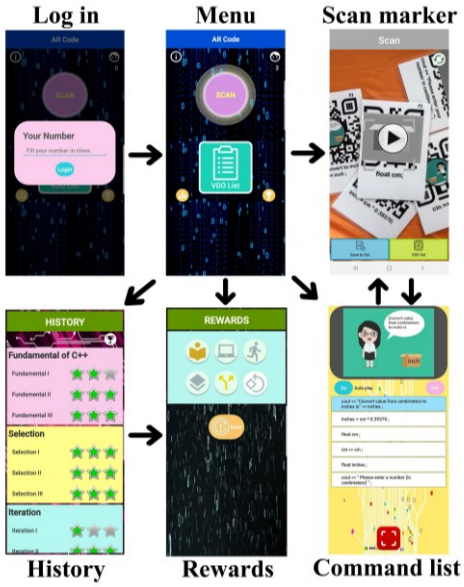
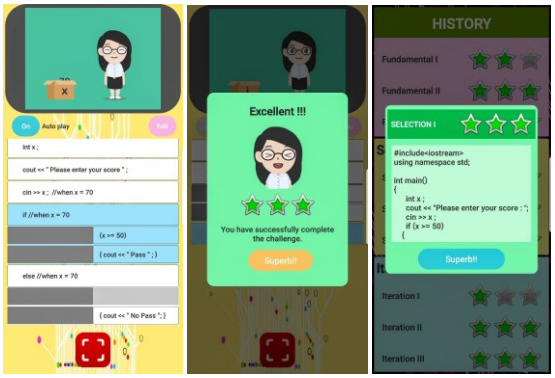


Figure 4. Application user interface

Figure 5. Animation clip overlaid and played on the screen

The code checker is activated after the last video in the command list is played. It checks whether the code order is correct, and identifies an incorrect order using rules specifically for each problem. Then the result windows will pop-up as shown in Fig. 6 b). If a user correctly completes the problem, the user will be given the full 3 stars. If there are some errors, stars will be deducted and the pop-up window will show a video telling what the user has missed. The username, stars, codes and topic name will be sent to our external server for retrieving user history and determining challenge rewards later.



a) b) c)

Figure 6. a) Command list, b) Result pop-up window, c) Detail for each topic history

We designed ARCode as a game to increase the motivation and reduce the fear of learning programming, by using History and Rewards. In Fig. 6 c), the History shows that a user had finished all problems, but not getting all three stars for every problem. Using the number of stars as a reward will make a user want to beat the game. For each problem, a user can re-check the codes that the user arranged before as shown in Fig. 6 c). The user can find possible improvement(s) and replay to achieve more stars to get additional stars. The number of finished problems and the number of received stars are calculated as various medals for a user to collect. For example, a user that earns all three stars for five problems will receive a silver medal for the Perfect Programmer medal, while if a user earns all three stars for eight problems will receive a gold medal. We plan to include the leaderboard for each kind of medal to further motivate and make the game more challenging and interactive among learners.

IV. EXPERIMENTAL METHODS

A. Populations

We have tested our system with students from the Faculty of Engineering, Khon Kaen University, who registered for an introductory computer programming class during October to November 2018. Most of students were the first and the second year students. We had 28 participants: random 14 participants in a control group and 14 participants in a treatment group.

B. Data Collection and Analysis

We used the contents from three elementary but essential topics of C++ programming: Fundamentals of C++ (basic program structure, input and output, elementary data types, and math and logical operations), Selection (conditions and selectively executing commands), and Iteration (conditions, reusing variables, and repeating tasks). For both groups, we conducted a paper-based pre-test consisting of coding problems in various difficulties. Then both groups had regular lectures for 2 weeks, during which we expected the control group to self-study from available documents and exercise outside

Figure 7. Learning environment of ARCode

of class. However, the treatment group had a supplemental session to use ARCode for 1 hour 30 minutes as shown in Fig. 7. After solving three problems for each topic, they filled out user satisfaction questionnaire. Finally, both groups took a paper-based post-test. We collected and analyzed the following data.

1) *Pre-test and post-test scores*: for analyzing how effective the system allowed users to develop programming skills. The full score was 10 for each topic and 30 in total. We applied t-test with 95% confidence level (p-value $\leq$ 0.05) on pre-test and post-test scores of control and treatment groups to learn the achievement differences.

2) *User satisfactions:* for evaluating the system as a motivating and collaborating tool in learning programming. We assessed user satisfactions by the questionnaire referenced from Technology Acceptance Model 3 (TAM3) [20]. There were 29 questions of seven factors: opinion toward IT and AR, system usefulness, enjoyment, ease of use, context controllability, user interaction, and adoption of ARCode.

3) *User performance and behavioral data from ARCode:* This included results of the game play, the number of stars, the number of user trials, and user codes. Not only this data helped us improve the application, but behavioral data could also indicate how users were engaged to learning programming with ARCode.

V. RESULTS AND DISCUSSIONS

A. System Effectiveness for Learning Programming

a) Achievement analysis between pre-test and post-test scores of the control group

From Table I, the average scores of the post-test for the control group increased from the pre-test by around 1 point or less for all topics. The improvements were also not significant. However, the standard deviations of the post-test were much less than the those of pre-test (more density of scores around the higher mean scores). This indicates that, regardless of method, the more you learn and practice on programming, the more chance you can improve on the skills.

b) Achievement analysis between pre-test and post-test scores of the treatment group

In Table II, the average post-test overall score of the treatment group significantly increased by 2.43 points. When looking at individual topics, however,

only the average score for the Fundamentals of C++ increased significantly although by less than 1 points. This could be because the average pre-test score was already high at 9.07 points; so there could not be much improvement. Nevertheless, the standard deviation is much lower at 0.27, indicating that most participants in the treatment group achieved almost perfect post-test scores in this topic.

The average score for Iteration increased the most by 1.25 points, although just marginally significantly. The average score for Selection increased very little and not significantly. This could be because the latter two topics required more logical thinking practice. Plus, there might be insufficient time for participants to interact with the system since there were only 1:30 hours for both learning to use the system and interacting with the system. Nonetheless, when comparing the control and the treatment groups in Table I and II, we found that the treatment group's average scores increased more in all topics except for Selection, and increased significantly for the overall score. The scores of the control group did not increase much nor significantly. These results showed the potential of our system to help programming learners achieve their goals.

c) Achievement analysis of pre-test score between the control and treatment group

Table III shows that the pre-test scores of the treatment group were slightly higher than those of the control group, but not significantly. Therefore, in this experiment, the prior knowledge of participants in the two groups did not differ significantly.

d) Achievement analysis of post-test score between the control and treatment group

Comparing the post-test scores between the two groups in Table IV, the treatment group performed significantly better in all topics and overall scores, except for Selection. Given that the pre-test scores were not significantly different between the two groups, these results indicated the positive potential of using our system to help improve programming skills, which were consistent with the results in section a) and b).

In Selection and Iteration topics, the standard deviations of the treatment group's post-test were a lot higher than those of the control group. It indicated that there were students that both performed highly and yet lowly in the treatment group. Moreover, in the Selection topic, the post-test scores of two groups were very close (8.04 vs 8.05) and not significantly different. This could be because the complication of Selection and Iteration topics that may need more time to practice to be proficient. To confirm these observations and causes, we will need to modify our experiment in the future, by having longer data collection time, a larger number of participants, and better quality of the participants (e.g. paying more attention). Finally, we need to improve and verify our tests to better reflect the programming skills of participants given the limited test time.

TABLE I. COMPARISONS OF AVERAGE PRE-TEST AND POST-TEST SCORES FOR THE CONTROL GROUP

Topic	Pre-test		Post-test		p-value
	$\bar{x}$	S.D.	$\bar{x}$	S.D.	
Fund. of C++	8.30	5.47	8.33	3.63	0.48
Selection	7.10	9.02	8.04	1.99	0.13
Iteration	3.72	8.81	3.85	4.83	0.41
Total	19.12	45.32	20.21	12.30	0.23

TABLE II. COMPARISONS OF AVERAGE PRE-TEST AND POST-TEST SCORES FOR THE TREATMENT GROUP

Topic	Pre-test		Post-test		p-value
	$\bar{x}$	S.D.	$\bar{x}$	S.D.	
Fund. of C++	9.07	1.58	9.78	0.27	0.03*
Selection	7.57	12.16	8.05	8.26	0.15
Iteration	4.95	9.22	6.20	11.69	0.07**
Total	21.60	49.49	24.03	36.08	0.02*

* is significant ($p \leq 0.05$), ** is marginally significant ($p \leq 0.10$)

TABLE III. COMPARISONS OF AVERAGE PRE-TEST SCORES FOR THE CONTROL AND TREATMENT GROUPS

Topic	Control		Treatment		p-value
	$\bar{x}$	S.D.	$\bar{x}$	S.D.	
Fund. of C++	8.30	5.47	9.07	1.58	0.29
Selection	7.09	9.02	7.57	12.16	0.70
Iteration	3.72	8.81	4.95	9.22	0.29
Total	19.12	45.32	21.60	49.49	0.35

TABLE IV. COMPARISONS OF AVERAGE POST-TEST SCORES FOR THE CONTROL AND TREATMENT GROUPS

Topic	Control		Treatment		p-value
	$\bar{x}$	S.D.	$\bar{x}$	S.D.	
Fund. of C++	8.32	3.63	9.78	0.27	0.01*
Selection	8.04	1.99	8.05	8.26	0.49
Iteration	3.85	4.83	6.20	11.69	0.02*
Total	20.21	12.30	24.03	36.08	0.03*

* is significant ($p \leq 0.05$), ** is marginally significant ($p \leq 0.10$)

B. *Feasibility Analysis of the System as a Motivating and Collaborating Learning Tool*

Table V shows the levels of user satisfactions in all seven aspects. We found that 90.63% of users had high attitude (level 4 and 5) toward using IT and AR. Although some did not experience with AR before, 86.25% had foreseen that AR could help increase motivation for learning, and help improve their programming skills. It was not surprised that this young generation was ready to embrace new technology. The results were consistent with Teng and Chen [11], showing that AR could be potentially effective learning tool.

There were 80.21% of users agreeing that the system was useful. In particular, when asking whether the animations helped them understand the operations of C++ commands, 75.31% of users highly agreed. There were 83.33% of users agreeing that they enjoyed using the system. More specifically, 81.25% highly agreed that the game made them have fun in learning. Moreover, ARCode could reduce the fear of programming as 87.50% of users had agreed so.

TABLE V LEVEL OF SATISFACTION AFTER USING ARCODE APPLICATION AND ENVIRONMENT

Topic	Percentage of users with different levels of satisfaction (%)				
	5	4	3	2	1
Opinion toward IT and AR	34.38	56.25	9.37	0.00	0.00
Usefulness	23.96	56.25	19.79	0.00	0.00
Enjoyment	39.58	43.75	16.67	0.00	0.00
Ease of use	16.67	47.92	35.42	0.00	0.00
Context Controllability	41.67	51.04	7.29	0.00	0.00
User interaction	18.75	75.00	6.25	0.00	0.00
System Adoption	31.25	43.75	21.88	3.13	0.00

5 is most satisfied and 1 is least satisfied

According to the application log, many users kept trying to solve problems to get all three stars. These results indicated that our system had a high potential to both help learners better understand the lessons and help user enjoy and dare to learn programming.

Nevertheless, only 64.59% of users highly agreed with the ease of use of the system, although 92.71% of participants thought they could control and use all functions easily and fully. The low satisfactions on the ease of use could come from the delay of marker recognition, raising complaints from users. This was due to both environmental conditions (light, phones, etc.) and the application bugs. Therefore, we still need to improve the scanning function of the application, including the marker design.

We designed the learning environment to be collaborative. Users could share markers, but used their own device to collect and order commands. Since they could exchange knowledge and experience from each other, 93.75% of participants were highly satisfied with the user interaction aspect of our system. In the future, we can have learners to interact with each other within the online application, so that they can learn together anywhere and anytime.

Finally, 75% of users highly agreed on the future adoption and recommendation of ARCode. Users perceived ARCode as a promising system for learning programming, which was useful, enjoyable, easy to use and control, and promoting collaboration. This report confirmed that we were on the right track and encouraged us to keep improving the system.

VI. CONCLUSIONS

We proposed ARCode, the system aiming to help novice programmers with technical and motivational problems. The system employs AR, game, and collaboration to help learners with the systematic thinking, while providing enjoyment, challenge, and sharing environment to motivate learners.

We primarily found that users improved significantly overall after using our system. Yet we still need to improve experimental methods and our system, including adding new motivating features like online collaboration. However, this work showed that the system can potentially support novice programmers on technical and motivational issues.

REFERENCES

[1] Code.org. Code.org - Learn on Code Studio [Internet]. [cited 2017 Oct 29]. Available from: https://studio.code.org/courses

[2] K. Chaipah and S. Sittiyuno, "The Study of Engineering Students' 3-Step Computer Programming Process." Proceedings of the 2nd STEM Education Conference (iSTEM-Ed 2017), Chiang Mai, Thailand, July 12-14, 2017.

[3] A. Moreno, M. S. Joy, "Jeliot 3 in a Demanding Educational Setting." Electronic Notes in Theoretical Computer Science, vol. 178, pp.51-59 Jul 2007.

[4] C++ Tutor - Visualize C++ code execution to learn C++ online [Internet]. [cited 2017 Jun 23]. Available from: http://www.pythontutor.com/cpp.html#mode=edit

[5] code2flow - online interactive code to flowchart converter [Internet]. [cited 2017 Aug 27]. Available from: https://code2flow.com/app

[6] Visualize Python, Java, JavaScript, TypeScript, and Ruby code execution [Internet]. [cited 2017 Aug 8]. Available from: http://www.pythontutor.com/visualize.html#mode=edit

[7] Scratch - Imagine, Program, Share [Internet]. [cited 2017 Dec 6]. Available from: https://scratch.mit.edu/

[8] C. Kazimoglu, M. Kiernan, L. Bacon, L. MacKinnon, "Learning Programming at the Computational Thinking Level via Digital Game-Play." Procedia Computer Science, vol. 9, pp.522-531, 2012.

[9] L. Del Bosque, R. Martinez, J. L. Torres, "Decreasing Failure in Programming Subject with Augmented Reality Tool." Procedia Computer Science, vol. 75, pp.221-225, 2015.

[10] C-H. Teng, J-Y. Chen, "An Augmented Reality Environment for Learning OpenGL Programming." in IEEE, pp.996–1001, 2012.

[11] L.A. Tychonievich, Java Program Flow Visualizer [Internet]. [cited 2017 Jun 23]. Available from: https://www.cs.virginia.edu/~lat7h/cs1/JavaVis.html

[12] M. Moisala, V. Salmela, L. Hietajärvi, S. Carlson, V. Vuontela, K. Lonka, et al., "Gaming is related to enhanced working memory performance and task-related cortical activity". Brain Research, vol. 1655, pp.204-215, Jan 2017.

[13] C. Kazimoglu, M. Kiernan, L. Bacon, L. Mackinnon, "A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming." Procedia - Social and Behavioral Sciences, vol. 47, pp.1991-1999, 2012.

[14] G. Fessakis, E. Gouli, E. Mavroudi, "Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study." Computers & Education, vol. 63, pp.87-97, Apr 2013.

[15] G. A. Gunter, R. F. Kenny, E. H. Vick, "Taking educational games seriously: using the RETAIN model to design endogenous fantasy into standalone educational games." Educational Technology Research and Development, vol. 56(5-6), pp.511-537, 2008.

[16] A. Wakatani, T. Maeda, "Automatic generation of programming exercises for learning programming language." in Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on. IEEE, pp.461-465, 2015.

[17] R. Y. Tahboub, J. Shin, A. Abdelsalam, et al., "LIMO: learning programming using interactive map activities." in ACM Press, pp.1–4, 2015.

[18] K. J. Harms, N. Rowlett, C. Kelleher, "Enabling independent learning of programming concepts through programming completion puzzles." in Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on. IEEE, pp.271-279, 2015.

[19] Vuforia. Vuforia Developer Portal [Internet]. [cited 2019 Mar 29]. Available from: https://developer.vuforia.com /

[20] V. Venkatesh, H. Bala, "Technology acceptance model 3 and a research agenda on interventions." Decision Sciences, vol. 39(2), pp.273-315, May 2008.