Active Learning through Game Play in a Data Structures Course

Darina Dicheva
Computer Science Dept.
Winston-Salem State University
Winston Salem, NC 27110, USA
dichevad@wssu.edu

Austin Hodge
Computer Science Dept.
Winston-Salem State University
Winston Salem, NC 27110, USA
ahodge114@rams.wssu.edu

ABSTRACT

Data Structures is a fundamental Computer Science discipline, challenging students' abstract thinking, problem solving and programming skills. In this paper, we present an educational game intended to explicate several features hindering students' understanding of the data structure Stack on conceptual and practical level. The game targets all three aspects of teaching data structures: conceptualization, application and implementation. These aspects are embodied as three parts of the game tied together through a meaningful storyline. The application part targets the use of stacks to solve problems, such as converting arithmetic expressions from infix to postfix notation and evaluating postfix and infix expressions. The implementation part involves solving Parson's problems and writing Java code for implementing the methods of the Stack class. The results of the conducted evaluation of the game show statistically significant learning gains for the students and a strong positive attitude towards this type of active learning.

CCS CONCEPTS

• **Applied computing** → **Education**; *Computer-assisted instruction*

KEYWORDS

Educational games, data structures, stacks, active learning

ACM Reference format:
D. Dicheva and A. Hodge. 2018. Active Learning through Game Play in a Data Structures Course. In *SIGCSE'18: 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA.* ACM, NY, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159605

1 INTRODUCTION

Data Structures, one of the core Computer Science courses, is concerned with providing students with understanding of abstract data types: their specification, implementation, performance characteristics, and application. The course typically covers the primary data structures stacks, queues, lists, trees, maps, and graphs. Data Structures is a conceptually demanding subject, which confronts a significant proportion of students early in the course. It has a well-founded conceptual basis, where students must simultaneously build and apply several higher order cognitive skills in order to solve a problem, which often turns out to be difficult for many. To alleviate this difficulty, there has been extensive work on creating tools for supporting data structures learning mainly through visualization [1]. However, a recent study [2] has shown that simple visualization is not enough to support successful learning of algorithmic processes. Students may look at dynamic visualizations without understanding the context and deeper meaning. When students are engaged in a learning activity, such as exploring, experimenting or predicting, they learn better.

The nature of computing and the generation of students have changed drastically in the past years. Still, most higher education computing courses are taught in traditional ways that may not be adequate to keep up with the changing educational reality and may not support the necessary learning [3]. Developing computing skills at higher learning levels, including the ability to apply the learned concepts in practice, requires using active learning and motivational strategies. Active learning is especially important in introductory Data Structures courses that teach fundamental conceptual and programming constructs [4].

Educational games, which aim to produce learning outcomes or attain educational objectives, are a form of active learning, intended to promote student engagement and contribute to student learning based on learners' own experiences. Game-based learning is considered a promising instructional method and is believed to result in a wide range of benefits, including increased learning effectiveness, interest, motivation and persistence. In the last two decades educational games have emerged as an evolving instructional strategy for computing education. Driven by the need to provide more hands-on opportunities for computing students, various educational games have been created to stimulate the learning process in different computing disciplines [5, 6]. Most of these games however concentrate on teaching topics from a relatively small number of disciplines with special focus on software engineering, introductory programming and algorithms and complexity. A literature review on educational games for teaching computing in higher education published in 2016 [5] points out that of the 107 games identified by the authors, only 3 have been designed particularly for teaching data structures. This number is negligible compared to the 41 games created for software engineering. Similarly, [7] reports only 9 publications that

SIGCSE'18, February 21–24, 2018, Baltimore, MD, USA

ACM ISBN 978-1-4503-5103-4...$15.00
DOI: https://doi.org/10.1145/3159450.3159605

describe either educational games or game-based learning environments targeting data structures. We are aware of only 3 games which target the Stack data structure [8, 9, 10]. The first two target several data structures but provide only simple, interactive visualization. In [8] the user can insert and remove elements in a stack (through push and pop buttons); [9] offers a visualization of the Tower of Hanoi, as well as a Stack ship game, where the player must insert falling numbers in a stack with a shape of a spaceship, so as to match a given number sequence. In the "Tower of cubes" game [10] the player switches between a Stack and Queue mode when adding new cubes to the tower, since if two consecutive cubes have the same color, they disappear. This however is somewhat conceptually confusing since the same tower as considered as either a stack or a queue.

The lack of educational tools that support active learning of data structures and specifically stacks motivated us to develop the Stack Game – an educational game that provides an enjoyable and engaging interactive environment for learning stacks. The game was implemented by the second author, who is an undergraduate student and was used in two offerings of a Data Structures course.

This paper describes our approach to the design and implementation of the Stack Game, as well as the results of its use as an intervention in a regular Data Structures course. The conducted empirical study contributes to the body of knowledge on the effectiveness of game-based learning as an instructional strategy for engaging students in computing education.

2 OVERVIEW OF THE GAME

While on the first sight the concept of stack seems easy and intuitive, some students have difficulty to relate the simple real-world examples of stacks of trays or books to the abstract concept and its structural and operational features. This makes it difficult for them to envisage the use of this data structure for solving problems, which is of critical importance to the Computer Science field. Consequently, our goal was to create a game that targets the three main learning objectives associated with the Stack topic in a typical Data Structures course: understand the abstract concept of stack (the logical relationship among its data elements and the operations that can be performed on the structure), learn how to implement the structure and the operations (e.g. in Java), and learn to use stacks to solve problems. With relation to the latter, we selected popular problems, such as evaluating infix and postfix arithmetic expressions and converting expressions from postfix to infix notation. We designed the Stack Game in three parts each corresponding to one of the above learning objectives and tied them together using a meaningful storyline to promote engagement and active learning.

In the game, the player controls a robot that is traveling back home after completing a mission. The robot's spaceship however crashes and he has to walk, while passing through various obstacles/doors, which are to be unlocked by solving stack-related problems. Each part of the game includes several levels with increasing difficulty of the tasks.

2.1 The Stack Game: Part 1

Part 1 of the game helps the student understand the concept of stack by solving a puzzle, which is a simpler variant of the Tower of Hanoi puzzle. The player is given several colored blocks stored in an initial stack. The goal is to arrange them in a particular order (matching a given color combination) in a target stack, by using an additional *work stack*. In the game, the targeted colored block combination is the key for unlocking a door: by inserting the correct key in the *key stack* beside a door, the robot can unlock it.

There are four levels in this part of the game, each with a more difficult target key. The target key in the first level consists of a single colored block since the goal is for the player to understand how to play the game. The player can move the robot from one location to another by indicating a target point on the path.

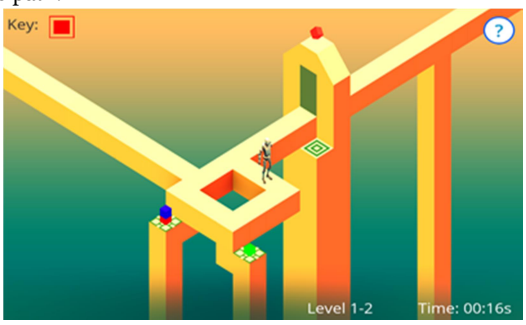

Figure 1: The Stack Game: Part 1 Level 2.

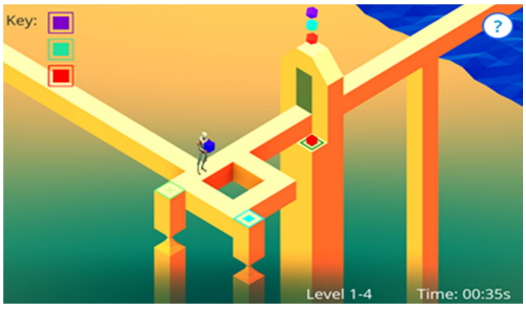

Figure 2: The Stack Game: Part 1 Level 4.

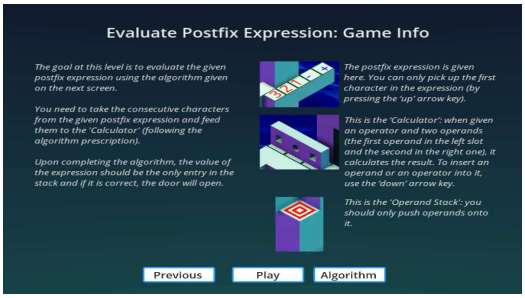

Figure 3: The Stack Game: an Introductory Screen.

When the robot is beside a stack, pressing the 'up' or 'down' arrow keys causes 'popping' or 'pushing' a block from/onto the stack. A popup bubble shows the name of the corresponding operation. In Fig. 1, the key is a red block. The robot is half way through the level and must pick up (pop) the purple block from

the left *stack* then proceed to push it onto the *work stack* on the right. This will leave the left stack with only one red block, thus allowing the robot to pick it up on the next move; and carry and push it onto the *key stack* to open the door and complete the level. In the first two levels, the player can see the content of each stack, while in the next two levels they can see only the top of the initial and work stacks (see Fig. 2). This provides an extra level of difficulty and gives the player a better understanding of the concept of stack. While the goal and context of work are the same at these levels, the given colored blocks and keys are different.

2.2 The Stack Game: Part 2

Part 2 of the game targets the application of stacks and includes three levels. Here, the robot is challenged in six situations and in order to open the corresponding doors has to solve the following problems (by using stacks):

- Convert an arithmetic expression from an infix to postfix notation (Level 2.1 and Level 2.2).
- Evaluate postfix expressions (Level 2.3 and Level 2.4).
- Evaluate infix expressions (Level 2.5 and Level 2.6).

Before these levels, the student is given an introductory screen explaining the task and what tools the robot can use to solve it and giving links to the corresponding algorithm. For example, Fig. 3 shows such a screen for Level 2.2. It specifies that to solve the tasks at this level the robot can use an *Operand Stack* (for temporarily storing operands) and a *Calculator*, a device that calculates the result when given two operands and an operator. It also shows where the original arithmetic expression is and that in order to insert an operand or an operator into the calculator one should use the 'down arrow' key. Fig. 4 shows a state of the game play. The player has moved the robot to a position near the expression location, and using the 'up arrow' key has made it take the first symbol from the postfix expression. Since the picked up symbol is an operand, the player has to instruct the robot to go to the Operand Stack and push it there, then come back and pick up the next symbol from the expression. Fig. 5 shows another state of the game, where the robot has already taken the '+' operator from the expression, inserted it in the Calculator, popped an operand from the Operand Stack, inserted it in the left slot of the Calculator, popped a second operand from the stack and inserted it in the right slot of the Calculator. The screenshot shows that the Calculator has returned the result (the yellow block containing 3), which now has to be pushed onto the Operand Stack.

2.3 The Stack Game: Part 3

Part 3 targets stack implementation. Here, the robot has already reached the Robot's Kingdom. However, the main entrance of the fortress only opens when the four lamps in front of it get all lightened. This is a signal that a fellow robot is coming, since only robots know the secret of how to turn on the lights.

Each of the four lamps can be activated by submitting correct program code. The four lamps correspond to the main methods in the Stack class: pop(), push(), peek(), and isEmpty(). Entering the correct code for a method lightens up the corresponding lamp.

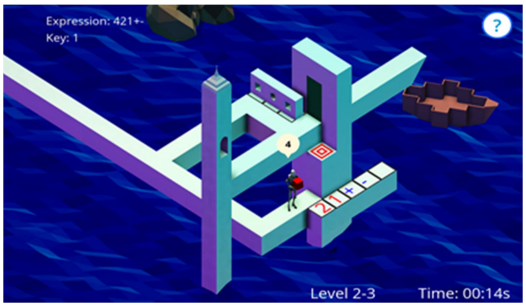

Figure 4: The Stack Game: Part 2 Level 3.

Figure 5: The Stack Game: Part 2 Level 3.

Figure 6: The Stack Game: Part 3 Level 1.

Figure 7: The Stack Game: Part 3 Level 4.

There are four levels in Part 3. Levels 1 and 2 offer Parson's problems (the statements are given but have to be arranged in the correct order through drag-and-drop functionality), while Levels 3 and 4 require writing missing pieces of code in a given method. Levels 1 & 3 target an array-based implementation of the stack (see Fig. 6) while Levels 2 & 4 target a linked-based

implementation (see Fig. 7). Which implementation is required at the particular level is clear from the information block in the upper left corner of the screen. This block displays the names of the instance variables of the class and method signatures. In Fig. 6 the robot has already entered correct codes for the methods push() and peek(), and is working on method pop(). If the player does not correctly complete the code, an error message will be given and she can try again.

3 GAME FEATURES

Game control and game log. The player is in control of both the gameplay and the game itself. They can pause or exit the game at any point of time and continue later from where they stopped. The game menu allows the player to replay any level already passed. However, the game is configured so that the player cannot skip a level and jump on a higher level without passing the lower ones. (This can be easily changed if necessary). All the interactions of the player are saved in a log with timestamps for further analysis. The logs are stored in an XML format.

Help and tutorial. The game includes a tutorial that can be accessed in three ways: (1) before playing the game: as a complete reading, providing all information needed for playing the game; (2) in the beginning of each of the three parts of the game: as an introduction to the corresponding part with references to the needed subject knowledge; (3) as help during the gameplay: information, relevant to the current task being handled. The information not only clarifies how to play the game at a particular level, but also provides the necessary subject knowledge, such as definitions, algorithms, etc., for user convenience.

Feedback and guidance. There are three types of feedback and guidance in the game. The first relates to indicating the correctness of the player's solution of the current problem, e.g. a door will open if the solution is correct and not otherwise. The second type is in the form of *constraints* imposed on the player so that they cannot perform particular incorrect actions in a particular game context, which will keep them on the correct track. For example, at Level 2-5, the player is not allowed to push a block carrying an operation onto the Operand stack. This is especially useful at the lower levels of the game, where the players are not very experienced. Moving to the higher levels, the constraints are gradually relaxed, leaving the responsibility of taking correct actions to the player. The third type of feedback is in the form of aggregated statistics of the player performance displayed on the Game Results page.

Timing. The completion of the tasks is not enforced. We believe that timing can be disadvantageous to students who have mastered the targeted skills but are not quick enough in the gameplay. Nevertheless, times are shown on the Game Results page, since the player can strive to improve their own performance. This can be motivational for returning back to the game.

The game has been developed by the second author using the Unity Game Engine.

4 EVALUATION

4.1 Using the Stack Game

The Stack Game was used in two consecutive offerings of a regular Data Structures course at a southern liberal arts college – in Fall 2016 and Spring 2017. Twenty nine students were enrolled in total (Fall 2016 – 10 students; Spring 2017 – 19 students), of which 76% were males and 24% were females. Of all students, 76% were African-American. The game was played as a learning activity in two regular classes/labs – the first one focused on the application of stacks and the second one on stack implementation. The course was taught by the same instructor (the first author), in the same format (flipped classroom) and using the same instructional materials (the textbook *Data Structures and Abstractions with Java* by F. Carrano & T. Henry). Since the course uses flipped learning, before both classes the students received an assignment requiring them to watch a 7-minute and 9-minute videos, explaining the material. They were also provided with reference to the corresponding content in the textbook and slides. In the beginning of the first lab, after a brief summary from the instructor, the students took a pre-test, then played Part 1 and Part 2 of the game and sent the log files to the instructor. In the second lab, they played Part 3 of the game, sent their log files to the instructor and took a post-test. They were also asked to take an online qualitative survey after the class.

The main purpose of the conducted evaluation was to evaluate the potential of the Stack Game as a tool supporting active learning, by assessing the knowledge gain of the students after using it. While all three parts of the Stack game were used in class, the knowledge gain of the students was assessed only after playing Part 1 and Part 2 (in the first lab), due to the lack of time (in the two designated labs) for accommodating a pre- and post-test on the material covered in Part 3 of the game.

The purpose of the pre-testing was to determine a baseline level of knowledge of selected stack concepts and skills prior to playing the game and to compare the results with the responses to the post-test to determine the level of student knowledge and skills after playing the game. For this reason, the questions in the pre-test and post-test were very similar. They included questions about drawing the content of a stack after a series of pops and pushes of elements, as well as converting infix arithmetic expressions to postfix notation, and evaluating postfix and infix expressions.

A secondary objective of the evaluation was to obtain information about students' opinion on the game as a learning tool and on the positive and negative aspects in terms of its motivational and educational power. The qualitative survey was related to all three parts of the game (see Table 3). It also included an open-ended question intended to capture students' general perceptions after the game play. The students whose data is reported here gave informed consent to participate in the study.

4.2 Results of the Evaluation

When considering the pre- and post-test, the pre-test scores ranged from a low of 0 to a high of 40 out of 40 points possible, with an average of 17.83 points, while post-test scores ranged

from a low of 15 to a high of 40 out of 40 points possible, with an average of 31.66 points (see Table 1 for means and standard deviation of results). Translating this to percentages, the pre-test had a range of 0% to 100% and the post-test a range of 37% to 100%.

Table 1: Pretest/Posttest Means

Assessment	Fall-16	Spring-17	Overall
Pre-test mean	17.00	22.53	17.83
Post-test mean	25.80	34.74	31.65
Pre-test StDev	12.95	9.55	11.05
Post-test StDev	12.19	8.12	10.43

T-tests were performed to determine whether the observed differences between the pre-test and post-test scores (the learning gain) are statistically significant. A standard way to assess the impact of the difference is to compute the effect size - the ratio of the difference in mean scores to the standard deviation. This indicates the magnitude of change in scores, in standard deviation units. A small effect size is considered less than 0.20, a medium effect size is 0.50, and a large effect size is 0.80 or greater. The large effect size shown in Table 2 indicates that the Stack Game had a substantial impact on students' knowledge gain.

Table 2: Effect Sizes along with T and P Values

Statistic Evaluation	Fall 16	Spring 17	Overall
T-test	2.0370	6.9915	6.2582
P value	0.0721	0.0000071	0.0000009
Effect Size	0.6996	1.3770	1.0320

Table 1 and Table 2 reveal also that while Spring-17 students earned the highest overall pre-test and post-test scores (means = 22.53, 34.74), they also earned the greatest gain scores (effect size = 1.377). These results suggest that using the game as part of the hands-on activities is a viable strategy for effective learning.

We also analyzed the gameplay logs for the proportion of the completed game levels and time spent on each level. The data indicates that 72% of the students completed successfully both parts of the game. There are no observable correlations between the time spent in the game and the post-test scores. However, the analysis of the gameplay logs suggests that those students who completed the two parts of the Stack Game were more likely to perform better on the post-test.

The questions of the conducted qualitative survey were aimed to capture students' judgment of the Stack game in six dimensions: perceived usefulness, perceived educational value, intention to use, clarity, provided support, and enjoyment. Participants were asked to evaluate to what extent they agreed with the statements in the questionnaire on a 5-point (Likert) scale from "Strongly Disagree" to "Strongly Agree". Table 3 shows the percent of positive answers ("Agree" and "Strongly Agree") from the 25 (out of 29) students who completed the survey. An open-ended question was also included in the survey aimed to obtain student responses on perceived positive and negative facets as well as constructive proposals for improvement of the game.

Table 3: Distribution of the Positive Answers (Agree and Strong Agree) to the Attitudinal Questions

Questions	Agree %
I enjoyed playing the game	84
The game helped me understand better the concept of stack	80
The game helped me understand better the use of stack for evaluating expressions	72
The game helped me understand better the implementation of the Stack Class	76
Instructions for Part 1 were clear	84
Instructions for Part 2 were clear	76
Instructions for Part 3 were clear	56
I played the game again after class	88
I will play again when preparing for Test 2	84
Game interface is intuitive	68
The help provided in Part 1 is sufficient	76
The help provided in Part 2 is sufficient	72
The help provided in Part 3 is sufficient	56
I wish to be given similar games for other topics covered in this course	92
I think educational games help clarifying subject concepts	96
Playing such games may increase my motivation to learn programing	96

The collected responses indicate a consistency among participating students, despite the varying level of their skills. They show that the majority of the students (96%) agreed that educational games help clarifying subject concepts. Very few students ("Strongly Disagree" - 4%, "Disagree" - 8%) claimed that the game did not help them to improve their understanding of the three aspects of stacks targeted by the game. Most of the students agreed that the Stack Game is a useful teaching tool and expressed an interest to play games targeting other data structures.

The perceived instructional quality of the game was supported also by the responses to the open-ended question "Any other comments about the game you would like to make". Examples of answers include:

- "The game was the best aspect of this course that i like so far. It helped me to understand the evaluating of arithmetic expressions very well."
- "I wish we have more games about every topic we cover in this course."
- "I believe hand on task help us understand the concept more. I am all for the game."

Some other answers pointed to places for possible improvements:

- "The directions for each level need to have bullets about the instructions of the game."
- "I think learning of the higher levels of the stack game need more assistance hints when students get stuck with the coding."
- "It would be a lot more helpful, if the programing code was to be found somewhere in the game."

We were looking also for some interesting findings relating the questionnaire and test results that can provide insights for future research. Specifically, students' responses to questionnaire items were mapped to their results in the post-test. One interesting observation emerging from this mapping is that students who responded with strong agreement to questions 2, 3 and 4 scored excellent results in the post-test. This raises the question of whether those students who believed that they benefited from the game would have scored lower without playing the game.

Our experiences with two cohorts backed by the evaluation results demonstrate that game-based learning in Data Structures context can enhance students' learning effectiveness and motivation. Although the class sizes were small, the differences in the scores were large enough to show both statistical significance and a large effect size, suggesting that the Stack Game is an effective learning tool. The accumulated experience opens the door to further studies based on that foundation.

5 CONCLUSIONS

Due to the difficulties that many students face in learning data structures, it is often problematic to impart working knowledge of their creation and operation by using only traditional teaching methods. As a response to these challenges, we designed the Stack Game. The objective was to enable students to acquire a correct cognitive model of the stack concept and behavior through gameplay. The Stack Game was incorporated as part of the learning activities in a regular Data Structures course, to promote deeper and more active learning. Our experience with designing and integrating the game into the classroom experience shows the importance of embedding sound instructional design into games in order to create effective learning tools. In particular, offering a game that includes all aspects of the targeted topic helps students to reflect on the learned concepts in a holistic way. The manner in which learning experiences from games are incorporated into the classroom is another important factor to its effectiveness. Instructors can further facilitate the transfer of skills by discussions connecting the game experience with relevant concepts students are learning in class.

Although there are many aspects in the instructional design to consider, learning outcomes are of critical importance for both the development and evaluation of educational games. Clarifying the goals and learning outcomes for a game not only provides a direction when designing it, but also provides metrics for evaluating the effectiveness of the created game. Our study results show that the participating students achieved statistically significant learning gains after they played the Stack Game. This suggests that playing the game helped them to develop a more correct and sound conceptual model for stacks and stack behavior. The responses to the questionnaire reveal that most of the students have positive attitude toward learning with the Stack Game and interest in using other similar games in learning Data Structures topics.

In the future, we plan to include better help for students when they are stuck in writing program code in Part 3 of the game. We will also develop a web-based version of the system, including automatic sending of the players' logs to the server.

ACKNOWLEDGMENTS

This work was partially supported by the NSF project HRD 1623236 "Increasing Student Motivation and Engagement in STEM Courses through Gamification".

REFERENCES

[1] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, et al. 2007. A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, 39(4), 204-223.

[2] C. Hundhausen, S. Douglas, and J. T. Stasko. 2002. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13(3), 259–290.

[3] G. Petri, C. G. von Wangenheim. 2017. How games for computing education are evaluated? A systematic literature review. *Computers & Education*. 107, 68-90.

[4] R. Lawrence. 2004. Teaching data structures using competitive games. *IEEE Transactions on Education*, 47(4), 459-466.

[5] P. Battistella and C. G. von Wangenheim. 2016. Games for teaching computing in higher education - A systematic review. *IEEE Technology and Engineering Education*, 9(1), 8-30.

[6] C. Johnson, M. McGill, D. Bouchard, M. Bradshaw, V. Bucheli, L. Merkle, M. Scott, Z. Sweedyk, J. Angel, Z. Xiao, M. Zhang. 2016. Game Development for Computer Science Education. Proceedings of the *2016 ACM Conference on Innovation and Technology in Computer Science Educatio*n (ITiCSE '17), Arequipa, Peru, 23-44.

[7] D. Dicheva, A. Hodge, C. Dichev, and K. Irwin. 2016. On the Design of an Educational Game for a Data Structures Course. *2016 IEEE Int. Conf. on Teaching, Assessment, and Learning for Engineering* (TALE), Bangkok, Thailand.

[8] N. Kaur and G. Geetha. 2015. Play and learn DS: interactive and gameful learning of data structure. *Int. J. Technology Enhanced Learning*, 7(1), 44–56

[9] E.B. Costa, A.M. Toda, M.A. Mesquita, J.D. Brancher. 2014. DSLEP (Data Structure Learning Platform to Aid in Higher Education IT Courses), *Int. Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, 8(4), 1143-1148

[10] B. Tan, J.L. Kim Seng. 2010. Game-based Learning for Data Structures: A Case Study, *2nd Int. Conference on Computer Engineering and Technology*, Chengdu, China, 718-721.