Uluslararası İleri Doğa Bilimleri ve Mühendislik Araştırmaları Dergisi
Sayı 7, S. 147-154, 5, 2023

Araştırma Makalesi

IJANSER
https://as-proceeding.com/index.php/ijanser
ISSN: 2980-0811

International Journal of Advanced Natural Sciences and Engineering Researches
Volume 7, pp. 147-154, 5, 2023

Research Article

Models of Data Structures in Educational Visualizations for Supporting Teaching and Learning Algorithms and Computer Programming

Ladislav Végh

Department of Informatics, J. Selye University, Slovakia

veghl@ujs.sk Email of the corresponding author

(Received: 06 June 2023, Accepted: 21 June 2023)

(1st International Conference on Pioneer and Innovative Studies ICPIS 2023, June 5-7, 2023)

ATIF/REFERENCE: Végh, L. (2023). Models of Data Structures in Educational Visualizations for Supporting Teaching and Learning Algorithms and Computer Programming. *International Journal of Advanced Natural Sciences and Engineering Researches, 7(5), 147-154.*

Abstract – Teaching and learning computer programming is challenging for many undergraduate first-year computer science students. During introductory programming courses, novice programmers need to learn some basic algorithms, gain algorithmic thinking, improve their logical and problem-solving thinking skills, and learn data types, data structures, and the syntax of the chosen programming language. In literature, we can find various methods of teaching programming that can motivate students and reduce students' cognitive load during the learning process of computer programming, e.g., using robotic kits, microcontrollers, microworld environments, virtual worlds, serious games, interactive animations, and visualizations. In this paper, we focus mainly on algorithm visualizations, especially on the different models of data structures that can be effectively used in educational visualizations. First, we show how a vector (one-dimensional array), a matrix (two-dimensional array), a singly linked list, and a graph can be represented by various models. Next, we also demonstrate some examples of interactive educational algorithm animations for teaching and learning elementary algorithms and some sorting algorithms, e.g., swapping two variables, summing elements of the array, mirroring the array, searching the minimum or maximum of the array, searching the index of minimum or maximum of the array, sorting elements of the array using simple exchange sort, bubblesort, insertion sort, minsort, maxsort, quicksort, or mergesort. Finally, in the last part of the paper, we summarize our experiences in teaching algorithmization and computer programming using algorithm animations and visualizations and draw some conclusions.

Keywords – Algorithms, Visualizations, Animations, Data Structures, Teaching And Learning, Computer Programming.

I. INTRODUCTION

Learning algorithms and programming are challenging for many computer science students. According to our research [1], 39% of undergraduate first-year computer science students at our university do not have any prior experience in computer programming. Novice programmers need to gain appropriate algorithmization skills, improve their logical thinking and problem-solving skills, learn the basic data types and data structures, and learn the syntax of a programming language. Because of the complexity and abstract nature of computer programming, it might be overwhelming for novice programmers; therefore, they might lose motivation after a few weeks of the introductory computer programming course. However, many

methods and tools might support teaching and learning computer programming and increase students' motivation [2]–[5].

II. VARIOUS METHODS AND TOOLS TO SUPPORT TEACHING AND LEARNING COMPUTER PROGRAMMING

Using educational robotic kits and robots in informatics education [6]–[10] shows an actual application of computer programming to novice programmers. Instead of seeing the output only on the monitor's screen, students can watch how a robot executes the instruction of the written program code, which has considerable motivation power.

Another possibility to motivate novice programmers is using games in computer programming education. Usually, every child likes playing games; even high school or university students like playing computer games sometimes. Therefore, students might learn basic computer programming concepts using serious games in teaching and learning computer programming [11]–[15]. After gaining some experience in programming, students might even develop their simple computer games [16]–[20].

By using microcontrollers in computer science education [21]–[26], students can see another practical application of programming in real life, which might increase novice programmers' motivation. Furthermore, students can use various sensors and switches as inputs and LEDs, LED panels, motors, and speakers as outputs, which could be exciting and motivating.

Using microworlds and microworld environments [27]–[30] in programming education might also be helpful for novice programmers to gain appropriate algorithmic thinking and basic programming skills playfully.

Another way to increase students' motivation in computer science education might be using virtual worlds [31]–[32] or simulations [33].

Finally, we would like to mention using visualizations of data structures and using interactive algorithm animations in programming education [34]–[39] that can help students understand some basic concepts of computer programming and algorithms. Moreover, after gaining programming experience, students can quickly develop visualizations and interactive animations, e.g., using JavaScript libraries [40].

III. EDUCATIONAL MODELS OF DATA STRUCTURES IN ALGORITHM VISUALIZATIONS

As we mentioned, using visualization and interactive animations in teaching and learning computer programming is one of the ways to motivate students and help them to understand data structures and algorithms effectively and efficiently. However, the education model representing the standard data structure is crucial to visualization or interactive animation. Using appropriate models, novice programmers can quickly process information and learn illustrated concepts or algorithms [35], [36]. In the following part of this section, we show how various educational models could represent a vector (one-dimensional array), a matrix (two-dimensional array), a linked, and a graph.

A. *Vector (one-dimensional array)*

Vector is usually one of the first standard data structures that novice programmers learn to use during introductory programming courses. For example, one of the most common ways to visualize a one-dimensional array is using a row of squares with numbers (Fig. 1). The indexes of the elements are shown above every square. Even though this way of visualization might be enough in many cases, for some students, it could be overwhelming because they need to process the information (the values of elements representing the numbers), and at the same time, they need to focus on the steps of the algorithm. To process the information more manageable, especially for young students, it might be better to use some graphical representation instead of a textual (numeric) representation [41].

0	1	2	3	4	5	6	7	8	9
5	12	7	12	19	1	20	8	3	14

Fig. 1 Visualization of a vector with numbers

One of the ways to illustrate a vector graphically is to use columns, where the values of the elements are represented by the heights of the columns (Fig. 2). This way of representing a one-dimensional array is advantageous when the algorithm compares the values of the elements, e.g., when sorting algorithms are demonstrated [37], [42], [43]. Furthermore, except for the heights of the columns (values of elements), the color of the columns could also be used to show some information, e.g., the

green color can illustrate the sorted part of the array, the red color can represent the unsorted part of the array, and the yellow color can show which two elements are compared or swapped (Fig. 2) [36].

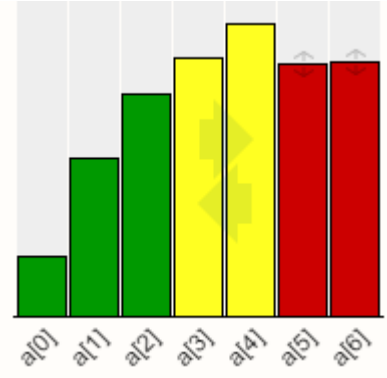

Fig. 2 Visualization of a vector with columns [44]

Another way to visualize a one-dimensional array is to use colorful balls, where the brightness of the color represents the value of the elements (Fig 3) [45]. This type of visualization helps demonstrate sorting algorithms, as well as the representation of the vector with columns.

Fig. 3 Visualization of a vector with colorful balls [45]

Using playing cards as a visualization of a one-dimensional array in the interactive demonstration of a sorting algorithm can also be motivating and engaging for novice programmers (Fig. 4) [43], [46].

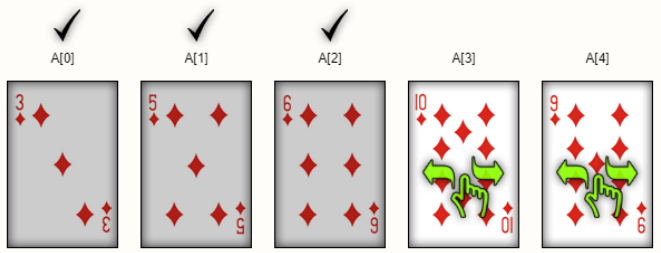

Fig. 4 Visualization of a vector with playing cards [46]

B. *Matrix (two-dimensional array)*

After gaining experience with one-dimensional arrays, students learn to use matrices (two-dimensional arrays). The most common way of visualizing a matrix is using squares with numbers in several rows and columns (Fig. 5). Before the first column are shown the indexes of the rows, and above the first row are shown the indexes of the columns (Fig. 5).

	0	1	2	3	4	5
0	6	11	7	13	10	1
1	13	5	15	7	5	14
2	18	8	12	2	19	4
3	3	16	6	20	9	5

Fig. 5 Visualization of a matrix with numbers

This way of representing a matrix is usually appropriate in most cases. However, suppose we want to emphasize the values of the elements, compare the values of some elements in the matrix, or show the location of the low or high values in the matrix. In that case, we can use colors to represent the values of elements in the matrix (Fig. 6). The bright colors can represent the high values of the elements, and the dark colors can represent the low values of the elements in the matrix.

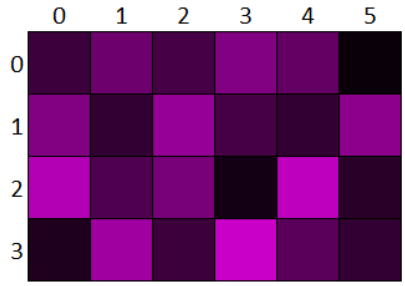

Fig. 6 Visualization of a matrix with colors (heatmap)

C. *Linked list*

Using dynamically created linked lists is usually one of the most challenging tasks in introductory programming courses. However, in our opinion and experiences, appropriate visualizations of dynamically created linked lists might help students to understand this topic easily and quickly [47], [48].

The visualization of a dynamically created linked list contains the representation of the pointers and the connected nodes in a similar way as they are stored in the memory of the computer (Fig. 7). In the image below, we can see two pointers (first, p2), which are pointing to the first and the last node of a singly linked list. In addition, the nodes contain a numeric value and a pointer to the next node (Fig. 7).

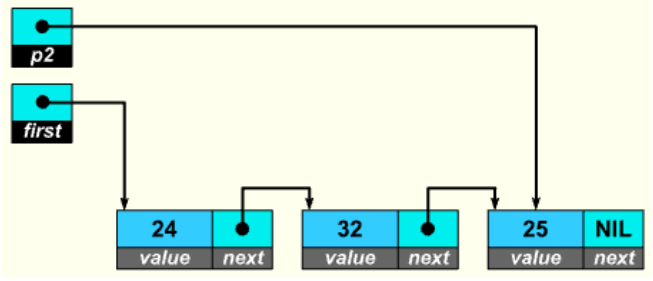

Fig. 7 Visualization of a singly linked list [49]

D. *Graph*

Automation of the visualization of graphs is usually a challenging task [34], mainly because a graph consists of several vertices (nodes) which could be connected with directed or not directed edges. To properly illustrate a graph, it is necessary to place the nodes in such locations in the visualization that the connection between the nodes can be clearly viewed. For example, in Fig. 8, we can see a visualization of an undirected graph consisting of five nodes and seven edges.

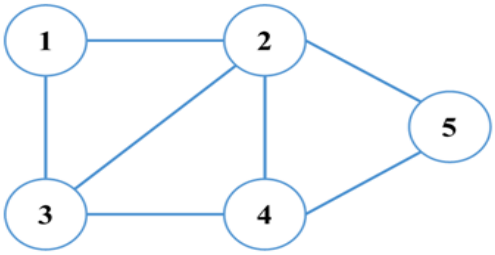

Fig. 8 Visualization of an undirected graph [50]

Furthermore, to use a graph in computer programs, novice programmers must understand that the programming language has no standard "graph" data structure. Instead, to represent a graph in a computer program, they must use a matrix (Fig. 9) or a list (Fig. 10).

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Fig. 9 Representing the graph with adjacency matrix [50]

The adjacency matrix (Fig. 9) consists of the same number of rows and columns. The number of nodes in the graph gives the number of rows and columns. When two nodes are connected in the graph, the intersection of the given row and column is number 1 in the adjacency matrix (otherwise, there is 0). For an undirected graph, every edge is represented twice in the adjacency matrix, e.g., the edge between nodes 2 and 3 is represented by value 1 in the 3rd column of the 2nd row and the 2nd column of the 3rd row [50].

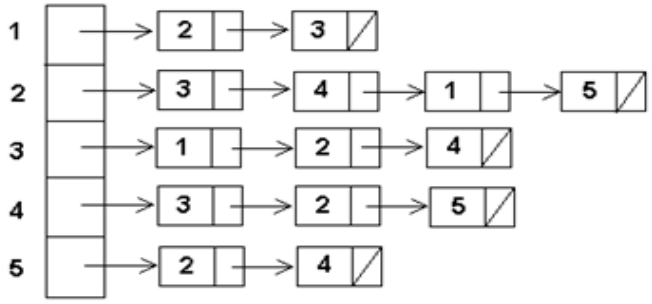

Fig. 10 Representing the graph with adjacency list [50]

The adjacency list contains a vector of pointers representing the nodes of the graph (vertical column in Fig. 10). Every element of the vector points to a dynamically created singly linked list, which contains the number of nodes that the given node is connected with [50].

IV. Examples of Algorithm Visualizations

Visualizations of data structures could be used in printed books, electronic studying materials, interactive animations, etc. [51]. However, students probably benefit the most from using interactive animations, where the visualizations show the changes dynamically according to the steps of the demonstrated computer science algorithms.

The first basic algorithms that students meet during the introductory programming courses are swapping two variables, summing elements of the array, mirroring the array, searching the minimum or maximum of the array, searching the index of the minimum or maximum of the array, etc. Examples of interactive animations demonstrating these elementary algorithms can be found on the webpage [52]. Fig. 11 shows a screenshot of an interactive animation illustrating the summing of elements of a one-dimensional array. Except for the visual representation of the one-dimensional array, the interactive animation contains the visual representation of a variable ("sum"), an index variable ("i") used in the for loop of the algorithm, and the pseudocode of the algorithm with the highlighted line of code that is just executed.

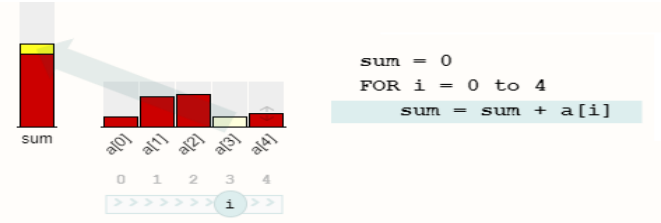

Fig. 11 Interactive animation demonstrating the summing of elements of a one-dimensional array [52]

Next, computer science students can learn sorting algorithms with time complexity $O(n^2)$, e.g., simple exchange sort, bubblesort, insertion sort, minsort, and maxsort. To introduce these sorting algorithms and learn the main steps, similarities, and differences of these sorting algorithms, students can use interactive animations of playing cards (Fig. 12) which can be found on the webpage [46]. The main benefit of using these animations with a conceptual view for introducing sorting algorithms is that students can see how the various sorting algorithms work without going into the details. Thus, there is a less cognitive load for students [43].

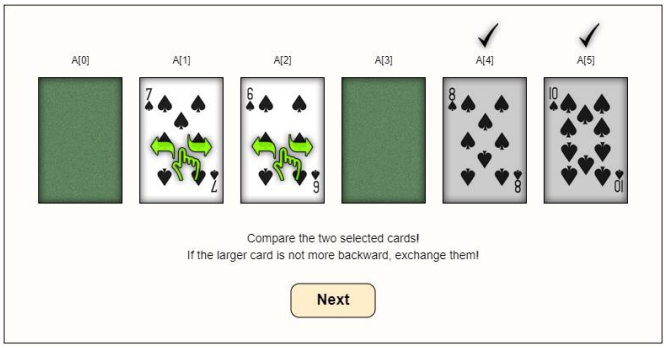

Fig. 12 Interactive animation of the bubblesort algorithm [46]

Similar demonstrations of sorting algorithms with a conceptual view can be found on the webpage [53], where robots sort colorful balls in ascending order from the darkest to the brightest using various sorting algorithms (Fig. 13).

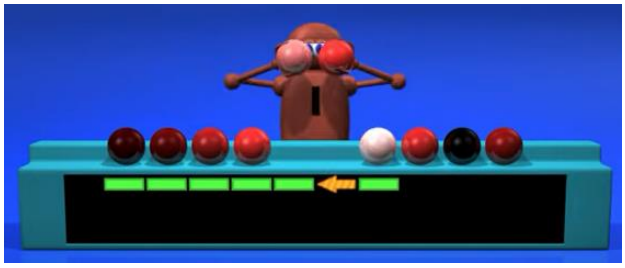

Fig. 13 Video demonstrating the insertion sort algorithm [45]

After understanding the main steps, similarities, and differences of these sorting algorithms using animations with a conceptual view, it is advised for students to use micro-level animations of the same sorting algorithms. The micro-level animations show the algorithms in more detail; the animations usually contain indexes of the elements, representations of loop variables, pseudocode, or program code with the highlighted lines of currently executed instructions (Fig. 14). Examples of these interactive animations of sorting algorithms can be found on the webpage [44].

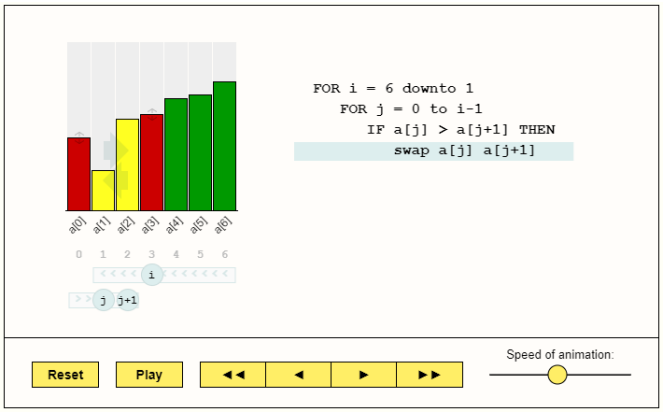

Fig. 14 Interactive animation of the bubblesort algorithm [44]

Students can step these interactive animations of algorithms forward or backward; they can change the input data, play the animation continuously, or change the speed of the animation. Using these animations, students understand the illustrated sorting algorithms in more detail. They see in the program code that these algorithms usually contain two nested loops and a conditional statement, and they fully comprehend how these algorithms work [43]. After understanding these simple sorting algorithms, later they can more easily understand recursive sorting algorithms with time complexity $O(n.\log_2 n)$, e.g., algorithms of quicksort or mergesort. Examples of interactive animations of such recursive sorting algorithms are available on the webpage [54].

Demonstrating the sorting algorithms with folk dances (Fig. 15) is another exciting and entertaining way of understanding these computer science algorithms [55], [56]. Fig. 15 shows a screenshot of a video illustrating the bubblesort algorithm with Hungarian ("Csángó") folk dance.

Fig. 15 Bubblesort algorithm illustrated with a folk dance [56]

V. Summary and Conclusion

In this paper, first, we described the possibilities of making algorithmization and computer programming more engaging, motivating, and playful for students using various methods and tools. Next, in the article's main part, we dealt with the educational models of data structures that can be used in visualizations. Finally, we showed some examples of interactive animations that can be used for teaching and learning programming.

According to our experiences and prior research [42], [43], [57], using appropriate interactive animations and visualizations can help students understand computer science algorithms easier and quicker, keep novice programmers motivated during the whole programming course, and make the learning algorithms more enjoyable.

Acknowledgment

The paper was supported by project KEGA 013TTU-4/2021 "Interactive animation and simulation models for deep learning".

References

[1] L. Végh and Š. Gubo, "Assessment of Algorithmic and Logical Thinking of First- and Second-Year Computer Science Students at J. Selye University in Academic Years 2019/20 and 2021/22," in *ICERI2022 Proceedings*, 2022, pp. 1888–1895. https://doi.org/10.21125/iceri.2022.0480

[2] T. Y. Gainutdinova, M. Y. Denisova, and O. A. Shirokova, "Current Trends in the Study of Object-Oriented Programming in Higher Education," in *Proceedings IFTE-2019*, 2019, pp. 163–169. https://doi.org/10.3897/ap.1.e0152

[3] J. Udvaros, N. Forman, and D. É. Dobák, "Application and impact of electronic solutions in teaching programming," *Annales Mathematicae et Informaticae, Special issue on Formal Methods in Informatics*, 2023. https://doi.org/10.33039/ami.2023.04.001

[4] K. Czakóová and V. Stoffová, "Training teachers of computer science for teaching algorithmization and programming," in *IMSCI'20 proceedings, The 14th International Multi-conference on Society, Cybernetics and Informatics*, 2020, pp. 231–235.

[5] J. Udvaros and O. Takáč, "Technical IT solutions in teaching," in *INTED 2022 Proceedings*, 2022, pp. 4047–4052. https://doi.org/10.21125/inted.2022.1107

[6] V. Stoffová and M. Zboran, "Educational Robotics in Teaching Programming in Primary School," in *Proceedings of International Conference on Recent Innovations in Computing (ICRIC 2022), Lecture Notes in Electrical Engineering (LNEE, volume 1001)*, 2023, pp. 669–682. https://doi.org/10.1007/978-981-19-9876-8_51

[7] V. Chaudhary, V. Agrawal, P. Sureka, and A. Sureka, "An Experience Report on Teaching Programming and Computational Thinking to Elementary Level Children Using Lego Robotics Education Kit," in *2016 IEEE Eighth International Conference on Technology for Education (T4E)*, 2016, pp. 38–41. https://doi.org/10.1109/T4E.2016.016

[8] K. Mohamed, Y. Dorgham, and N. Sharaf, "Kodockly: Using a Tangible Robotic Kit for Teaching Programming," in *Proceedings of the 13th International Conference on Computer Supported Education*, 2021, vol. 1, pp 137–147. https://doi.org/10.5220/0010446401370147

[9] A. Fegely and H. Tang, "Learning programming through robots: the effects of educational robotics on pre-service teachers' programming comprehension and motivation," *Educational technology research and development*, vol. 70, pp. 2211–2234, 2022. https://doi.org/10.1007/s11423-022-10174-0

[10] O. Takáč and J. Udvaros, "Implementation of the Principles of Obstacle Detection with the Help of Irobot Roomba in the Teaching of Computer Science," in *ICERI 2021 Proceedings*, 2021, pp. 6682–6687. https://doi.org/10.21125/iceri.2021.1509

[11] A. Akkaya and Y. Akpinar, "Experiential serious-game design for development of knowledge of object-oriented programming and computational thinking skills," *Computer Science Education*, vol. 32, issue 4, pp. 476–501, 2022. https://doi.org/10.1080/08993408.2022.2044673

[12] N. Bouali, E. Nygren, S. S. Oyelere, J. Suhonen, and V. Cavalli-Sforza, "Imikode: A VR Game to Introduce OOP Concepts," in *Koli Calling '19: Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, 2019, pp. 1–2. https://doi.org/10.1145/3364510.3366149

[13] E. Lotfi and M. Bouhorma, "Teaching Object Oriented Programming Concepts Through a Mobile Serious Game," in *SCA '18: Proceedings of the 3rd International Conference on Smart City Applications*, 2018, pp. 1–6. https://doi.org/10.1145/3286606.3286851

[14] D. Bundhoo and L. Nagowah, "Gaming with OOP Learn: A Mobile Serious Game to Learn Object-Oriented Programming," in *3rd International Conference on Next Generation Computing Applications (NextComp)*, 2022, pp. 1–6, https://doi.org/10.1109/NextComp55567.2022.9932243

[15] N. Singh and L. Nagowah, "OOP Codes: Teaching Object-Oriented Programming Concepts Through

Mobile Serious Game," in *25th International Computer Science and Engineering Conference (ICSEC)*, 2021, pp. 377–382 https://doi.org/10.1109/ICSEC53205.2021.9684593

[16] L. Végh and O. Takáč, "Teaching and Learning Computer Programming by Creating 2D Games in Unity," in *ICERI 2021 Proceedings, 14th International Conference of Education, Research and Innovation*, 2021. pp. 5696–5700. https://doi.org/10.21125/iceri.2021.1285

[17] V. Gabaľová, M. Karpielová, and V. Stoffová, "Beginners Online Programming Course for Making Games in Construct 2," in *Proceedings of International Conference on Recent Innovations in Computing (ICRIC 2022), Lecture Notes in Electrical Engineering (LNEE, volume 1001)*, 2023. pp. 547–558. https://doi.org/10.1007/978-981-19-9876-8_41

[18] K. Czakóová, "Game-based programming in primary school informatics," in *INTED 2021, Proceedings of the 15th International Technology, Education and Development Conference*, 2021, pp. 5627–5632. https://doi.org/10.21125/inted.2021.1134

[19] K. Czakóová and J. Udvaros, "Applications and games for the development of algorithmic thinking in favor of experiential learning," in *EDULEARN21 : Proceedings of the 13th International Conference on Education and New Learning Technologies*, 2021, pp. 6873–6879. https://doi.org/10.21125/edulearn.2021.1389

[20] K. Czakóová, "Developing algorithmic thinking by educational computer games," in *Proceedings of the 16th International Scientific Conference "eLearning and Software for Education": eLearning sustainment for never-ending learning*, 2020, vol. 1, pp. 26–33. https://doi.org/10.12753/2066-026X-20-003

[21] J. Udvaros and O. Takáč, "Developing Computational Thinking by Microcontrollers," in ICERI 2020 Proceedings, 13th annual International Conference of Education, Research and Innovation, 2020, pp. 6877–6882. https://doi.org/10.21125/iceri.2020.1474

[22] M. Csóka and K. Czakóová, "Innovations in education through the application of raspberry pi devices and modern teaching strategies," in INTED 2021, Proceedings of the 15th International Technology, Education and Development Conference, 2021, pp. 6653–6658. https://doi.org/10.21125/inted.2021.1327

[23] J. Udvaros and K. Czakóová, "Developing of computational thinking using microcontrollers and simulations," in EDULEARN21, Proceedings of the 13th International Conference on Education and New Learning Technologies, 2021, pp. 7945–7951. https://doi.org/10.21125/edulearn.2021.1619

[24] J. Udvaros and K. Czakóová, "Using teaching methods based on visualizing by TinkerCad in teaching programming," in ICERI 2021 Proceedings, 2021, pp. 5913–5917. https://doi.org/10.21125/iceri.2021.1333

[25] M. T. Fülöp, J. Udvaros, Á. Gubán, and Á. Sándor, "Development of Computational Thinking Using Microcontrollers Integrated into OOP (Object-Oriented Programming)," Sustainability, vol. 14, no. 12: 7218, 2022. https://doi.org/10.3390/su14127218

[26] J. Udvaros and L. Végh, "New teaching methods by using microcontrollers in teaching programming," in Proceedings of the 16th International Scientific Conference "eLearning and Software for Education", 2020, pp. 630–637. http://doi.org/10.12753/2066-026X-20-119

[27] K. Czakóová, "Microworld environment of small language as „living laboratory" for developing educational games and applications," in *Proceedings of the 13th International Scientific Conference „eLearning and Software for Education", Could technology support learning efficiency?*, 2017, vol. 1, pp. 286–291. https://doi.org/10.12753/2066-026X-17-042

[28] V. Stoffová and K. Czakóová, *Úvod do programovania v prostredí mikrosvetov*, Komárno, Slovakia: J. Selye University, 2016.

[29] M. Kölling, "The Greenfoot programming environment," *ACM Transactions on Computing Education*, vol. 10, no. 4, pp. 1–21, 2010. https://doi.org/10.1145/1868358.1868361

[30] S. Cooper, W. Dann, and R. Pausch, "Alice: A 3-D tool for introductory programming concepts," *Journal of Computing Science in Colleges*, vol. 15, no. 5, pp. 107–116, 2000.

[31] M. Turcsányi-Szabó, L. Csízi, and L. Végh, "Virtual Worlds in Education – best practice, design and research consideration," *Teaching Mathematics and Computer Science*, vol. 10, issue 2, pp. 309–323, 2012. https://doi.org/10.5485/TMCS.2012.0308

[32] L. Végh and M. Turcsányi-Szabó, "Using a Virtual School for Teaching and Learning the Basics of 3D Modeling and LSL Scripting in Second Life," in *eLearning and Software for Education: Could technology support learning efficiency?*, 2017, vol 1, pp. 572–579. https://doi.org/10.12753/2066-026X-17-084

[33] N. Annuš and O. Takáč, "Algorithmic and Simulation-Based Teaching of Computer Science and Mathematics in Higher Education," in *ICERI2022 Proceedings, 15th annual International Conference of Education, Research and Innovation*, 2022, pp. 4904-4911. https://doi.org/10.21125/iceri.2022.1184

[34] I. Bende, "Data Visualization in Programming Education," *Acta Didactica Napocensia*, vol. 15, no. 1, pp. 52–60, 2022. https://doi.org/10.24193/adn.15.1.5

[35] M. Esponda-Arguero, "Techniques for visualizing data structures in algorithmic animations," *Information Visualization*, vol. 9, issue 1, pp. 31–46, 2010. https://doi.org/10.1057/ivs.2008.26

[36] R. Fleischer and L. Kucera, "Algorithm animation for teaching," *Software Visualization, Lecture Notes in Computer Science*, vol. 2269, pp. 113–128, 2002.

[37] L. Végh, "Animations in Teaching Algorithms and Programming," in *Nové technologie ve vzdělávání*, Olomouc, Czech Republic, 2011.

[38] J. Udvaros and M. Gubán, "Demonstration the class, objects and inheritance concepts by software," *Acta Didactica Napocensia*, vol. 9, no. 1, pp. 23–34, 2016.

[39] J. Udvaros, "Teaching Object Oriented Programming by Visual Devices," in *Proceedings of the 15th International Scientific Conference "eLearning and*

Software for Education", 2019, vol. 1, pp. 407–413. https://doi.org/10.12753/2066-026X-19-054

[40] J. Udvaros and L. Végh, "Possibilities of Creating Interactive 2D Animations for Education Using HTML5 Canvas JavaScript Libraries," in *Proceedings of the 16th International Scientific Conference "eLearning and Software for Education"*, 2020, pp. 269–274. https://doi.org/10.12753/2066-026X-20-119

[41] R. E. Mayer, *Multimedia Learning*, 2nd ed., New York, USA: Cambridge University Press, 2009.

[42] L. Végh and V. Stoffová, "An interactive animation for learning sorting algorithms: How students reduced the number of comparisons in a sorting algorithm by playing a didactic game," *Teaching Mathematics and Computer Science*, vol. 14, no. 1, pp. 45–62, 2016. https://doi.org/10.5485/TMCS.2016.0415

[43] L. Végh and V. Stoffová, "Algorithm Animations for Teaching and Learning the Main Ideas of Basic Sortings," *Informatics in Education*, vol. 16, issue 1, pp. 121–140, 2017. https://doi.org/10.15388/infedu.2017.07

[44] L. Végh. (2020) Interactive animations for teaching algorithms and programming. Sorting algorithms 1. [Online]. Available: https://anim.ide.sk/sorting_algorithms_1.php

[45] udiprod. (2017) Insertion Sort vs Bubble Sort + Some analysis. [Online]. Available: https://youtu.be/TZRWRjq2CAg

[46] L. Végh. (2020) Interactive animations for teaching algorithms and programming. Sorting cards. [Online]. Available: https://anim.ide.sk/sortingcards.php

[47] L. Végh, "Elektronická podpora vyučovania dynamických údajových štruktúr," in *XIX. DIDMATTECH 2006*, Komárno, Slovakia, 2006.

[48] V. Stoffová and L. Végh, "Grafické modely dynamických údajových štruktúr a ich význam vo vyučovaní programovania," in *Trendy ve vzdělávání 2013*, Olomouc, Czech Republic, 2013.

[49] L. Végh. (2020) Interactive animations for teaching algorithms and programming. Singly linked list. [Online]. Available: https://anim.ide.sk/list.php

[50] T. Nagy and R. Giachetta. (2011) Gráfok ábrázolása. [Online], in I. Fekete and L. Hunyadvári (eds.) *Algoritmusok és adatszerkezetek*. Available: http://tamop412.elte.hu/tananyagok/algoritmusok/lecke23_lap1.html

[51] I. Bende, "Az algoritmus vizualizáció felhasználási lehetőségeinek vizsgálata az algoritmusoktatásban," PhD dissertation, Eötvös Loránd University, Budapest, Hungary, 2023.

[52] L. Végh. (2020) Interactive animations for teaching algorithms and programming. Basic algorithms. [Online]. Available: https://anim.ide.sk/basic_algorithms.php

[53] udiprod. Category: Computer Science. [Online] Available: https://www.udiprod.com/category/videos/computerscience/

[54] L. Végh. (2020) Interactive animations for teaching algorithms and programming. Sorting algorithms 2. [Online]. Available: https://anim.ide.sk/sorting_algorithms_2.php

[55] Z. Katai and L. Tóth, "Technologically and artistically enhanced multi-sensory computer-programming education," *Teaching and Teacher Education*, vol. 26, issue 2, pp. 244–251, 2010. https://doi.org/10.1016/j.tate.2009.04.012

[56] AlgoRythmics. (2011) Bubble-sort with Hungarian ("Csángó") folk dance. [Online]. Available: https://youtu.be/lyZQPjUT5B4

[57] L. Végh and O. Takáč, "Using Interactive Card Animations for Understanding of the Essential Aspects of Non-recursive Sorting Algorithms," in *Advances in Intelligent Systems and Computing: Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, 2017, pp. 336–347. https://doi.org/10.1007/978-3-319-46535-7_25