Automated Interactive Visualization on Abstract Concepts in Computer Science

Congmin Mao, Haoran Yu, Jiaxin Shi, Tingjun Cai, and Boyang Yang

***Abstract*—The paper presents CSVisFrame, a framework for making visualizations, which solves the understanding difficulty on learning abstract concepts in computer science including data structures and algorithms. With the framework, instructors and developers can develop all varieties of interactive visualizations, with which students can learn and understand abstract concepts in computer science more easily.**

CSVisFrame has been applied to both offline and online computer courses. Students from Sichuan Normal University have enjoyed visualizations based on CSVisFrame in their Algorithm Design and Analysis course, and thousands of students of Jisuanke have benefitted from online CSVisFrame-based visualized computer science courses. The effectiveness of CSVisFrame-based visualizations has been demonstrated by our sample survey, which shows that visualizations are widely accepted, and almost all students can achieve a better learning. CSVisFrame is open-sourced[1], and demonstrations based on CSVisFrame are free[2].

***Index Terms*—Visualization, Education, Framework, Computer Science.**

I. Introduction

COMPUTER science, a field that has a great deal of abstract concepts, for example, data structures, algorithms, computing models, communication protocols and relationships in software engineering, etc., is not easy to teach in a lecture only or reading only manner[1]. Instructors typically illustrate those concepts using pictures within slides or drawing sophisticated examples on the blackboard which are both prone to error. However, with the fast development of massive open online courses(MOOCs), students are spending more learning time on cyberspace instead of in traditional offline classrooms[2]. Developing effective online tools that reduce the comprehension difficulty of abstract concepts becomes crucial for learning computer science[3].

This paper presents a project called CSVisFrame that can be used to develop varieties of visualizations of computer science concepts. More than 20 visualization examples are developed based on the framework. Four main characteristics make CSVisFrame stand out of a vast number of related works in this field:

- Hierachical and extendable
- Cross-platform
- Interactive support
- Language independent

In all of above characteristics, interactive support is the most necessary characteristic to resolve the understanding difficulty on abstract concepts' learning. In CSVisFrame, interactive support is divided into three parts: data interactivity, operation interactivity, and display control interactivity. Data interactivity means that students can feed any inputs into abstract concepts, such as insert one or more given numbers into a BST(Binary Search Tree), and get visualized dynamic process as the feedback. For each data structure or algorithm, developers can support all of its operations easily with CSVisFrame, such as insertion, deletion and selection in a BST. Operation interactivity of CSVisFrame allows students to choose any operation as the next step, as well as some operations as a process list. To make the visualization be compatible with students in different learning speeds, CSVisFrame provides display control interaction, which allows students to control the visualized dynamic process like a handy video player.

To verify the learning effectiveness affected by CSVisFrame-based interactive visualizations, we visualized some data structures and algorithms. The visualizations are used both in an offline classroom and an online course. Students were requested to fill a survey after taking the course. Feedback of the survey suggested that the students learn abstract concepts better after using the visualizations.

The remaining part of this paper is structured as follows: Section II introduces related works. The design and implementations of the project are illustrated in Section III, IV. Section V introduces the types of courses that can be assisted by such a project. The outcome of teaching experiments is presented in Section VI. Section VII is the conclusion.

II. Related Work

As abstract concepts in computer science are difficult to understand, educators start developing visualizations as a tool since the 1980s. The "Sorting Out Sorting" video introduced in 1981 is the first visualization in computer science field that is ever known by people[4]. Yang, Shaffer and Heath developed a tool that animates C and C++ programs by calling on a Swan Annotation Interface Library(SAIL) in 1996, helping reduce the gap between algorithm runtime behavior and programs[5]. Mccallum-Rodney and Mugisa introduced a pseudocode construction animation software named PseudoCAS that can help novices understand programming structures[6]. Rainwater and Natarajan developed a Web-based project that allows students to learn algorithm courses online with animations in 2014 [7].

Congmin Mao, Haoran Yu, Jiaxin Shi and Boyang Yang are with Jisuan Institute of Technology, Beijing Judao Youda Network Technology Co. Ltd., China e-mail {maocongmin, yuhaoran, shijiaxin, yangboyang}@jisuanke.com

Cingjun Cai is with Cheriton School of Computer Science, University of Waterloo, Canada e-mail alex.cai@uwaterloo.ca

[1]CSVisFrame is available on github.com/Jisuanke/CSVisFrame.

[2]Demonstrations based on CSVisFrame are accessible at jisuan.tech/research/CSVisFrame/demo.

DOI: 10.5176/2251-3043_5.2.365

Since 2015, more comparative studies were coming out. Guo, White and Zanelatto developed a Web-based tool named Codechella that supposes multiple people can write code together and explore program visualization[8]. Wang et al. introduced a visualization tool named RBACvisual that can help students learning role-based access control[9]. Li et al. presented a tool to visualize and practice encryption and decryption process of Vigenère Cipher[10]. Dietrich, Goelman, Borror and Crook provided an animated courseware that helps students in different majors to understand relational databases[11]. Singh and Riedel devised a "growing library of AI data structures for visualization" that helps AI education in 2016[12].

Visualization tools have a positive impact on computer science education. Naps et al. explored the role of visualization in computer science education in 2002 and suggested that visualization technology has a positive influence on learning[1]. Halim implemented a tool called VisuAlgo in 2011 for algorithm and data structure visualization. It has been used during class in the National University of Singapore for many years, resulting in positive feedback[13]. A study introduced by Mulvey suggested that an algorithm visualization tool could enhance the teaching process and outcome in algorithm related courses[14].

III. Design Methodology

Different from most of the visualization projects, CSVisFrame is not a visualization tool for a single purpose. It is a framework that can be used to implement all varieties of visualizations. Unlike developing old-fashioned GIF images, Java components, and Adobe Flash animations, newly emerged web technologies become the main selections that support the visualization framework.

The goal of this work is to introduce a framework that supports instructions on abstract concepts and processes better than slides and drawing on the blackboard.

Features of related projects are studied, and the most common ones are supported in the framework. Four unique considerations below guide the design of the framework.

- Hierarchical and extendable: CSVisFrame is a framework with multiple layers that the lower three layers could be extended independently.
- Cross-platform: CSVisFrame is build up with cross-platform technologies, such as HTML, SVG, CSS, JavaScript. Any visualization made by using CSVisFrame can be used on desktop computers, laptops and hand-held devices with no further setup process.
- Interactive support: CSVisFrame can not only show the visualization. Students who use the CSVisFrame based visualization could also interact with them and understand the dynamic process how and why the visualization changes.
- Language Independent: As a framework, CSVisFrame is developed in a language independent manner. Specific visualization can be implemented with notes and explanations in any language.

IV. Implementation

A. Hierachical Design

The frame has four layers of hierarchic. The lower three layers are base components, combined components, visualization descriptions. The fourth layer is built based on the lower layers together with the demonstration player.

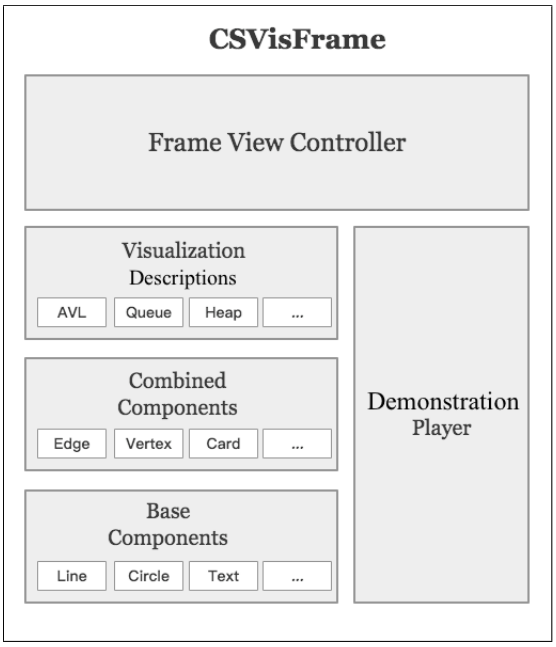

Fig. 1: CSVisFrame

Base components include classes and functions for drawing lines, Bézier curves, circles, rectangles, triangles and text information which are all derived from a class named "Base". Within the "Base" class, multiple methods, for example, copying, drawing and removing, that manipulate the elements on an SVG canvas are implemented.

Combined components include classes and methods for illustrating components made up of base components, for example, edges, vertices, multi-element boxes. These combined components share similar properties. The color, size, and text inside can be customized in the components of this layer.

Visualization descriptions layer is built of the combined components. On this layer, each visualization is described as a series of states containing the combination of combined components.

Together with the demonstration player, any designed visualization can be played on a canvas provided in frame view controller. The interface of interactions triggers can be added by providing input boxes and buttons. Who interact with the visualization can use the interface to generate states that illustrate the process of some changes on objects in the visualization. A player that controls the process of showing frames of states is also provided. People can use it to play, pause, resume, go to the next frame, go to the previous frame, jump to the last frame, jump to the first frame according to the needs of learning. Besides, pseudo codes and instructions could be displayed aside from it. Please see Fig. 3.

B. Extendable Structure

CSVisFrame is extendable on lower three layers. Any base component can be derived from the "Base" class. For example, the class named "Circle" is a subclass of the "Base". Properties that related to the newly defined base component are then described as variable members of the class, for instance, componentName, tagName and defaultProps. The properties described can be used by the display related methods to display the components finally. Besides, to hide the components accordingly when needed, a "hide" method should be defined for all extended components.

All combined components have independent classes which are derived from a shared class named "Composite". If a new combined component is added, within the subclass, it will define which base components to use and how it may change the position.

At the visualization description layer, adding any new visualization will need to derive a subclass from "Algorithm" class. A constructor should be added to initialize an initial state which will be duplicated and changed to generate new states later. Any possible manipulations can be defined in the class as methods which will create new states related to the process of visualization.

C. Visualization with CSVisFrame

To implement an interactive visualization of an abstract concept, relevant concepts and logics of the data structure should be added first. Then implemented components should be bound to the data structure. In addition, the initial state of the visualization should be created in the last step. Afterward, logics related to the visualization defined in the first step is visualized by generating new states consist of components. Lastly, fit states into the canvas and display them, and generate buttons with input boxes to control the animation of visualization. Please see Fig. 2.

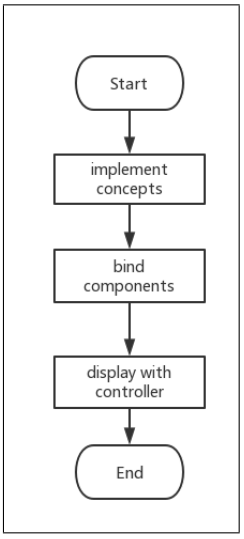

Fig. 2: The Frame

Any visualization of on abstract concept can be added by following the above procedure. A particular example is introduced in the next subsection.

D. Example: AVL Tree

In this subsection, AVL tree is illustrated as a visualization example. For the sake of implementing an interactive visualization of an AVL tree[15], a self-balanced binary search tree structure, a data structure for storing an AVL tree needs to be implemented first. A "Node" class is programmed for representing the nodes in the AVL tree structure. The member variables and methods of the Node class are added then. Using the "Node" class, an AVL tree is introduced with methods including "singleRightRotate", "singleLeftRotate", "doubleLeftRotate", "doubleRightRotate", "getSuccessor", "getPrecursor", "searchElement", "travLevel", "insertElement", and "removeElement".

Combined components, including "Vertex" and "Edge", are used to generate an initial state of the AVL tree structure. The coordinates of vertices are specified. Edges are described by using pairs of vertices.

After that, operations on AVL tree are illustrated. When the tree is being operated, "duplicateState" function is called to generate a new state, modify the coordinates of vertices, and appends the new state to a list of states.

Upon completion of the above work, the initial state is then displayed on a canvas. Buttons and input boxes are added for triggering "insert", "remove" and "search" operations. After any operations being triggered, new states are added to the states list and could be displayed on canvas one by one if needed.

V. Course information

CSVisFrame, a framework that supports the visualizations of abstract concepts in the field of computer science, can be used in all types of courses, including courses related to data structures, algorithms, computing models, communication protocols, etc.

In this study, algorithm related visualizations, for instance "AVL Tree" and "Binary Search Tree", are implemented based on CSVisFrame. These visualizations were adopted by "Algorithm Design and Analysis" course, a course for sophomore students offered by College of Computer Science, Sichuan Normal University, China. This course covers the knowledge of data structure and fundamental algorithms. Students are required to understand the knowledge and be able to implement data structures and algorithms accordingly after learning.

Besides, visualizations implemented by CSVisFrame are also adopted by Jisuanke, an online computer science education platform[16], as an embedded module. It is used by all students who registered "CS 261: data structure" course on the platform. In this course, the concepts of data structures and dynamic processes of related algorithms are highlighted. Visualizations implemented with CSVisFrame, for example "Queue", "Stack", "Heap", "Size Balanced Tree" and "Prim Algorithm", are used in the course.

VI. Student survey and discussion

For the sake of understanding how visualizations implemented with CSVisFrame contributes to learning outcomes,

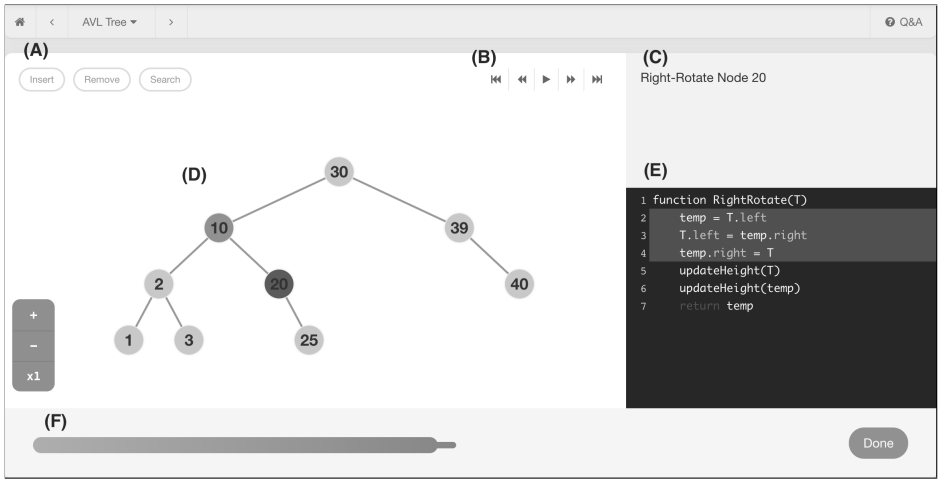

Fig. 3: The visualization made based on CSVisFrame typically has the following parts: (A)operation interface with "insert", "remove", "search" that trigger state of visualzations, (B)player that allows fast-backwarding, backwarding, playing, pause, forwarding and fast-forwarding on states (C)optional explanation of a state (D)the canvas that holds the AVL visualization, (E)corresponding pseudo code of a state, (F) optional progress bar of a player.

surveys are sent to both students who study relevant courses in Sichuan Normal University and on Jisuanke platform.

The survey consists of two parts, five evaluation questions, and a free response question. The evaluation questions can be responded as 7: strongly agree, 6: agree, 5: somewhat agree, 4: neutral, 3: somewhat disagree, 2: disagree, 1: strongly disagree. 23 valid forms are collected from students major in computer science and technology, Sichuan Normal University. 75 valid forms are gathered from randomly selected online learning students on Jisuanke platform.

TABLE I: Survey Questions

Q1	Visualization is more intuitive for me
Q2	I think interactive visualization is better than animation
Q3	Visualization assists me in understanding
Q4	I understand the data structure better after using the visualization
Q5	The visualization is hard to use and worthless
Q6	How you judge the visualization compare to traditional methods?

A. Statistical Analysis

The following results from students of Sichuan Normal University and online students studying on Jisuanke are collected.

TABLE II: Result from students of Sichuan Normal University

Question	Average Score
Q1	6.35
Q2	6.52
Q3	6.18
Q4	5.69
Q5	1.92

According to the result, the majority of students believe that interactive visualizations of abstract concepts are useful for their understanding and the interactive visualizations developed based on CSVisFrame are intuitive.

However, as an overall outcome evaluation, some students believe the visualizations do not have significant impacts on their understanding level, and a small part of students

TABLE III: Result from online students studying on Jisuanke

Question	Average Score
Q1	6.28
Q2	6.28
Q3	6.07
Q4	5.61
Q5	1.46

expresses the unsatisfying opinion on the visualization. From post-analysis investigation, it is found that the majority of those students are special cases since they already master the knowledge well before using the visualizations.

B. Students Comments

The feedback about visualizations made based on CSVisFrame are positive in general. Some students stated "The data structure and algorithm visualizations are very helpful. It deepens my understanding of the knowledge. I love the user interface as well. It is perfect.", "The visualization shows the dynamic processes of changes in data structures. The visualization of rotation on the tree structure is particularly helpful.", "The single step mode is very useful. Control it according to my understanding ability is preferable to play the animation directly.". At the same time, some issues were discovered by the students: (1) to show the instructions for previous and next step, together with the current instruction might help the understanding; and (2) the pseudo-code is a little bit far away from the visualization canvas which makes it hard to follow up.

Some students compared the visualization to the traditional instruction. They prefer visualization demonstrations over the traditional instruction methods. A student mentioned "It helps a lot. I was completely confused after the AVL lecture in my college. The visualization on the website is a lot better than my instructor.", "I would recommend it to my instructor. The visualization is very cool. Hope my classmate could have a chance to use it in class."

However, some students give suggestions for possible improvement: (1) "It could be better if shortcuts on the keyboard can be added to control the visualization demonstration.", (2) "The color in the visualization could be enhanced for better visual experience."

Combining the survey results and the comments, we are confident to assert that the learning experience and outcome of abstract concepts learning are improved by visualizations made by CSVisFrame.

VII. Conclusion

The study introduced a framework CSVisFrame which can be utilized to make visualizations for abstract concepts in computer science. Visualizations of data structures and algorithms are presented as examples. In addition, they were adopted in the real world teaching experiments both online and offline classroom. Students can engage in interaction with the visualizations and better understand the concepts.

According to the outcome of the survey, the visualizations made by CSVisFrame are effective in helping students obtaining abstract concepts in particular fields. The overall outcome is positive, and the students highly appreciate the visualizations made with CSVisFrame.

As suggested in the feedback, shortcuts on the keyboard will be added, and the framework will support more context information about current instruction. Besides, the CSVisFrame is opened to the public so that more visualizations can be added by contributors all over the world. The source codes of CSVisFrame are available on github.com/Jisuanke/CSVisFrame, and demonstrations are accessible at jisuan.tech/research/CSVisFrame/demo.

Acknowledgment

The authors would like to thank the students of Sichuan Normal University and Jisuanke who participated in the study and filled in the survey.

References

[1] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger *et al.*, "Exploring the role of visualization and engagement in computer science education," in *ACM Sigcse Bulletin*, vol. 35, no. 2. ACM, 2002, pp. 131–152.

[2] P. Kim, *Massive open online courses: the MOOC revolution*. Routledge, 2014.

[3] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards, "Algorithm visualization: The state of the field," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 3, p. 9, 2010.

[4] J. Stasko, *Software visualization: Programming as a multimedia experience*. MIT press, 1998.

[5] J. Yang, C. A. Shaffer, and L. S. Heath, "Swan: A data structure visualization system," in *International Symposium on Graph Drawing*. Springer, 1995, pp. 520–523.

[6] C. S. McCallum-Rodney and E. Mugisa, "Reducing abstractness of basic programming structures for novice programmers-using pseudocas," in *International Conference on Infocomm Technologies in Competitive Strategies (ICT). Proceedings*. Global Science and Technology Forum, 2014, p. 75.

[7] S. B. Rainwater and V. S. Natarajan, "The effectiveness of animations in teaching recursive algorithms," in *International Conference on Infocomm Technologies in Competitive Strategies (ICT). Proceedings*. Global Science and Technology Forum, 2014, p. 80.

[8] P. J. Guo, J. White, and R. Zanelatto, "Codechella: Multi-user program visualizations for real-time tutoring and collaborative learning," in *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on*. IEEE, 2015, pp. 79–87.

[9] M. Wang, J. Mayo, C.-K. Shene, T. Lake, S. Carr, and C. Wang, "Rbacvisual: A visualization tool for teaching access control using role-based access control," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2015, pp. 141–146.

[10] C. Li, J. Ma, J. Tao, J. Mayo, C.-K. Shene, M. Keranen, and C. Wang, "Vigvisual: A visualization tool for the vigenère cipher," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2015, pp. 129–134.

[11] S. W. Dietrich, D. Goelman, C. M. Borror, and S. M. Crook, "An animated introduction to relational databases for many majors," *IEEE Transactions on Education*, vol. 58, no. 2, pp. 81–89, 2015.

[12] S. Singh and S. Riedel, "Creating interactive and visual educational resources for ai," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[13] S. HALIM, "Visualgo–visualising data structures and algorithms through animation," *OLYMPIADS IN INFORMATICS*, p. 243, 2015.

[14] M. Mulvey, "Effects of visualization on algorithm comprehension," 2015.

[15] M. AdelsonVelskii and E. M. Landis, "An algorithm for the organization of information," DTIC Document, Tech. Rep., 1963.

[16] Jisuanke. Accessed: Jul. 24, 2016. [Online]. Available: http://www.jisuanke.com

Authors' Profile

Congmin Mao was born in 1993. He received the B.E. degree in computer science and technology major from Sichuan Normal University. He is interested in algorithms and data structures.

Haoran Yu was born in 1992. He received the B.S. degree with highest distinction in Mathematics and Computer Science from the University of Illinois at Urbana-Champaign, in 2015. His research interests include social computing, social visualization, computer science education, human-computer interaction.

Jiaxin Shi was born in 1994. He is a Ph.D. candidate in computer science and technology major, Tsinghua University. His research interests in knowledge graph, natural language processing.

Tingjun Cai was born in 1995. His is an undergraduate student in computer science major, University of Waterloo. His main research interests are software engineering, natural language processing.

Boyang Yang was born in 1991. He received the B.E. degree in computer science and technology major from Beihang University in 2013. His research interests include formal methods, programming languages, and computational advertising.