CHAPTER 1

INTRODUCTION

Algorithm Visualizer (also referred as algorithmic animation) uses dynamic graphics to see computation of a given algorithmic program. It made an attempt to make the algorithms robust during 80’s (Brown, 1988; Brown and Sedgewick, 1985), and also the golden age of algorithmic visualization was around the year 2000, when outstanding software tools for a dynamic algorithmic visualization (e.g., the language Java and its graphic libraries) and sufficiently powerful hardware were already available on the market. It had been expected that algorithmic visualization would replace the way algorithms are taught. Many algorithmic animations had appeared, largely for straight forward problems like basic tree data structures and sorting. There have been even attempts to automatize development of animated algorithmic program and algorithmic visualization. Another direction was to develop tools that might permit students to make their own animations simply. Rather than giving explicit references to algorithmic animation papers, the reader is directed to a super-reference (Algoviz) that brings a listing over 700 authors, a number of them even with twenty-nine references in algorithmic animation and visualization. There are also various web pages that supply algorithmic animation systems, e.g. Algoanim, Algomation, DD2, AlgoLiang, VisuAlgo. We powerfully believe that the rationale is relative simple: An algorithmic program operates on some information (the input variables, and also the output data). Usually, in any explicit field of engineering, there is a customized way of visualization of data - graphs and trees are drawn as circles connected by line segments, number sequences could be visualized as collections of vertical bars, there are standard ways of drawing matrices, vectors, real functions, etc. An algorithmic animation is sometimes executed by running the algorithmic program slowly or in steps, and easily modifying the visual illustration of the information within the screen. A person who understands the algorithm in question can see how the algorithmic program progress, but a beginner user simply see visual objects moving and bartering their shapes and colours, but finding out why the motion in picture show runs means sometimes too trouble for him or her . The solution that we provide is to see as what the algorithmic programme is doing, however why it’s operating in the way it is running. In alternative words, our aim is to see an abstract algorithmic idea that is in back of a certain computing method. The objective of this study is to style a system of algorithmic visualization, implement the system and visualize the run time for every implemented algorithmic programme aims to assist students acknowledge algorithms additional effectively.

CHAPTER 2

LITERATURE SURVEY

In this literature survey there are many existing systems that are aiming for algorithm visualization. Below are the few works that are considered for the survey:

Naps, T. L. [1] has developed Jhavé: Supporting algorithm visualization. JHAVE fosters the use of algorithm visualization as an effective pedagogical tool for computer science educators, helping students to better understand algorithms. The Java-hosted algorithm visualization environment (JHAVE) is not an AV system itself but rather a support environment for a variety of AV systems (called AV engines by JHAVE).

Karavirta, V., & Shaffer, C. A. [2] has developed a Creating engaging online learning material with the jsav javascript algorithm visualization library. The JavaScript AV development library. JSAV goes beyond traditional AV library support for displaying standard data structures components, to provide functionality to simplify creation of AVs on many engagement levels including interactive exercises. We describe the growing body of content created with JSAV and summarize our three years of experience and research results from using JSAV to build content that supports CS education.

Simonák, S. [3] has developed Algorithm visualization using the vizalgo platform. The paper contains concise overview of development in the area of software visualization, as well as a proposal of software visualization platform with the emphasis on its extensibility and portability.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 SOFTWARE REQUIREMENTS

Functional Requirements:

a. Visual Representation: The visualizer should accurately represent the algorithm's components (e.g., nodes, edges, arrays) and operations (e.g., swaps, comparisons) to provide a clear understanding of the algorithm's behaviour.

b. Step-by-Step Execution: The visualizer should execute the algorithm in a step-bystep manner, highlighting each step and updating the visual representation accordingly.

c. Interactive Controls: Users should be able to interact with the visualizer, such as pausing, stepping through the algorithm, modifying input data, adjusting algorithm parameters, or switching between different algorithms for comparison.

d. Multiple Algorithm Support: The visualizer should support various algorithms, allowing users to choose and visualize different algorithms based on their needs.

e. Performance Analysis: The visualizer may provide performance metrics, such as time complexity and space complexity, to enable users to analyze and compare the efficiency of different algorithms.

User Interface Requirements:

a. Intuitive Interface: The visualizer should have a user-friendly and intuitive interface that is easy to navigate and understand.

b. Customization Options: Users should have options to customize the input data, adjust algorithm parameters, and control the visualization settings to suit their specific needs.

c. Clear Visualization: The visual representation should be visually appealing, clearly depicting the algorithm's execution and highlighting key steps or elements.

d. Responsive Design: The visualizer should be responsive and accessible across different devices and screen sizes, ensuring a consistent user experience.

Performance Requirements:

a. Responsiveness: The visualizer should respond quickly to user interactions, providing a seamless and smooth experience.

b. Scalability: The visualizer should handle large input data sets and complex algorithms efficiently, without significant performance degradation.

c. Resource Optimization: The visualizer should optimize resource usage (e.g., CPU, memory, graphics) to ensure optimal performance on a variety of devices.

Compatibility Requirements:

a. Cross-Browser Compatibility: The visualizer should work consistently across different web browsers, such as Chrome, Firefox, Safari, and Edge.

b. Platform Compatibility: The visualizer should be compatible with different platforms, such as desktops, laptops, tablets, and mobile devices.

Documentation and Support:

a. User Documentation: Provide clear and comprehensive documentation on how to use the visualizer, including instructions, features, and troubleshooting guides.

3.2 HARDWARE REQUIREMENTS

**Operating system -** Windows 8 or later

**Processor -** Intel Pentium 4 or later

Memory - 2 GB minimum, 4 GB recommended

**Screen resolution -** 1280x1024 or larger

**Application window size -** 1024x680 or larger

**Internet connection -** Required

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

An algorithm visualizer typically consists of multiple layers or components that work together to provide a user-friendly and interactive way to understand how algorithms work. Here's a high-level overview:

Frontend Interface: This is the user-facing part of the visualizer. It includes the user interface (UI) elements that allow users to interact with the visualized algorithms. The frontend typically includes components like code input areas, visualization canvas, control buttons (play, pause, step), algorithm selection options, and parameter settings.

Backend Logic: The backend logic handles the core functionality of the algorithm visualizer. It's responsible for interpreting the user's input, running the algorithm step by step, and providing the necessary data for visualization. This component includes the algorithm implementations themselves and the logic for controlling the visualization process.

Visualization Engine : The visualization engine is responsible for rendering the algorithm's behaviour in an understandable and interactive way. It translates the algorithm's execution into visual representations that can be displayed on the UI. This might involve animations, graphs, charts, and other graphical elements to show how data structures and values change during each step.

User Interaction Management: This component manages user interactions, such as clicking on control buttons, adjusting parameters, and selecting different algorithms. It communicates with the backend logic to trigger algorithm execution and updates the visualization accordingly.

4.2 USE CASES

Education : Algorithm visualizers are extensively used in educational settings to help students understand complex algorithms. They provide a visual representation of algorithm execution, making it easier for students to grasp concepts like sorting, searching, and graph algorithms.

Interview Preparation : Job seekers preparing for technical interviews can use algorithm visualizers to practice coding challenges. Visualizing how algorithms work step by step can help in understanding and solving problems more effectively.

Visualizing Sorting Algorithms : Sorting algorithms, such as bubble sort, insertion sort, and quicksort, can be visually represented to demonstrate how elements are rearranged during sorting.

Graph Algorithms : Visualizers help in understanding graph algorithms like breadth-first search (BFS) and depth-first search (DFS), showcasing how these algorithms traverse graphs.

Algorithm Complexity Visualization : Complexities like time complexity and space complexity can be challenging to grasp. Visualizers can illustrate how these complexities change as input sizes grow.

CHAPTER 5

IMPLEMENTATION

5.1 METHODOLOGY

In this project, we aim to develop an effective algorithm visualizer tool that can engage students to remain on platform and improve learning outcomes. To achieve this goal, we analyse the pedagogy, usability and accessibility goals of the online students and incorporate the features of the above goals to design effective user interactions and visualizations for an online algorithm visualizer tool. In this design process, we involve three basic activities:

Establishing requirements

Designing alternatives and prototyping

Evaluation

In the establishing requirements activity, we conducted a comprehensive survey on the existing literature, which determines the goals and their relevant features. At the designing alternatives and prototyping activity, we designed the user interactions and the visualizations, where we followed the Schneiderman’s eight golden rules of user interaction design . In this work, we mainly highlighted the establishing requirements and designing alternatives and prototyping activities. The evaluation activity is currently in work-in progress, and it will be done based on a set of Likert survey questionnaire.

A diagram of a software development

Description automatically generated

Fig 1.1

The proposed system includes the visualization of algorithms such as sorting, Searching and path-finding algorithms. HTML5 and CSS are used for interface. HTML5 communicates with JavaScript code and vice versa to visualize the particular algorithm and update the interface accordingly as shown in above Fig with bidirectional arrow

A computer screen with a diagram and arrows

Description automatically generated with medium confidence

Fig 1.2

5.2 DATA FLOW

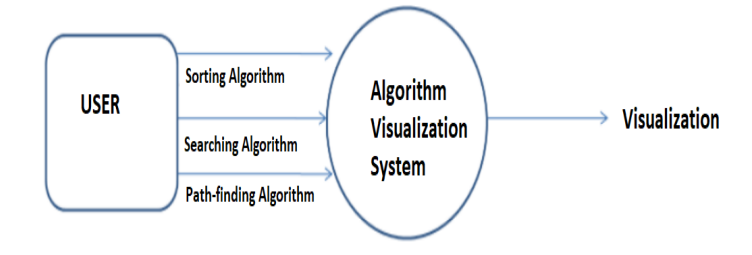


Fig 2.1

Data Flow Diagram in which rectangle present External entity (an outside system that sends or receives data) and circle show a Process (process that changes the data, producing an output). The arrows towards the process shows input while the arrows away from the process shows output.

A diagram of search engine optimization

Description automatically generatedFig 2.2

The proposed system contains 3 types of algorithms 1. Sorting Algorithm 2. Searching Algorithm 3. Path-finding algorithms

Sorting Algorithms: A sorting algorithm is used to rearrange the array or list of numbers according to a comparison operator on the elements. List of elements can be arranged in ascending or descending order as per comparison operator.

Searching Algorithms: Searching algorithms are developed to check or retrieve an element from a data structure where it is stored. These algorithms are classified in 2 main types based on the type of search operation Linear search: In this algorithm the list of array is traversed sequentially and every element is checked. Interval Search : This algorithm is specially developed to search in a sorted list of elements. It is more efficient since it does not check all the elements Example: Binary search

Path-finding Algorithm : There are many problems in computer science that needs user to find the shortest path between set points to solve such problems Path-finding Algorithms are developed.

CHAPTER 6

RESULTS AND DISCUSSION

Through a survey conducted by us we inferred that 60% of the students responded better to understanding concepts through visualization rather than their own imagination or the regular teaching methods.

It's been proved time and again through different experiments and research on masses that any kind of visual aid such as an image, a video or even an animation clip tends to be remembered more by humans.

Not only will the visualization help but due to features of mazes and patterns in our application, the students can relate the working of the algorithms to real life examples ( likes obstructions in the form of walls).

Often we see teachers struggling to make students understand concepts such as algorithms without it getting monotonous, that's where our project comes into play as a great teaching aid. Because of our user friendly and engaging interface the problem of distraction or losing interest tends to decrease, making it very efficient.

Our project can easily be incorporated alongside our education system by promoting different ways of learning rather than the age old blackboard method as we just need to access a website hosted on the internet to use the application.

And with uncertain times like nowadays, we cannot only afford to be dependent only on our teachers and one to one offline teaching to understand different concepts. E-learning is the new age learning technique and our project is a step towards reinforcing this method of learning.

6.1 SNAPSHOTS

Fig 3.1- User Interface -Home Page

A screenshot of a computer

Description automatically generated

Fig 3.2 -Sorting Visualizer

A screen shot of a video game

Description automatically generated

Fig 3.3 – Pathfinding Visualizer

A screenshot of a computer

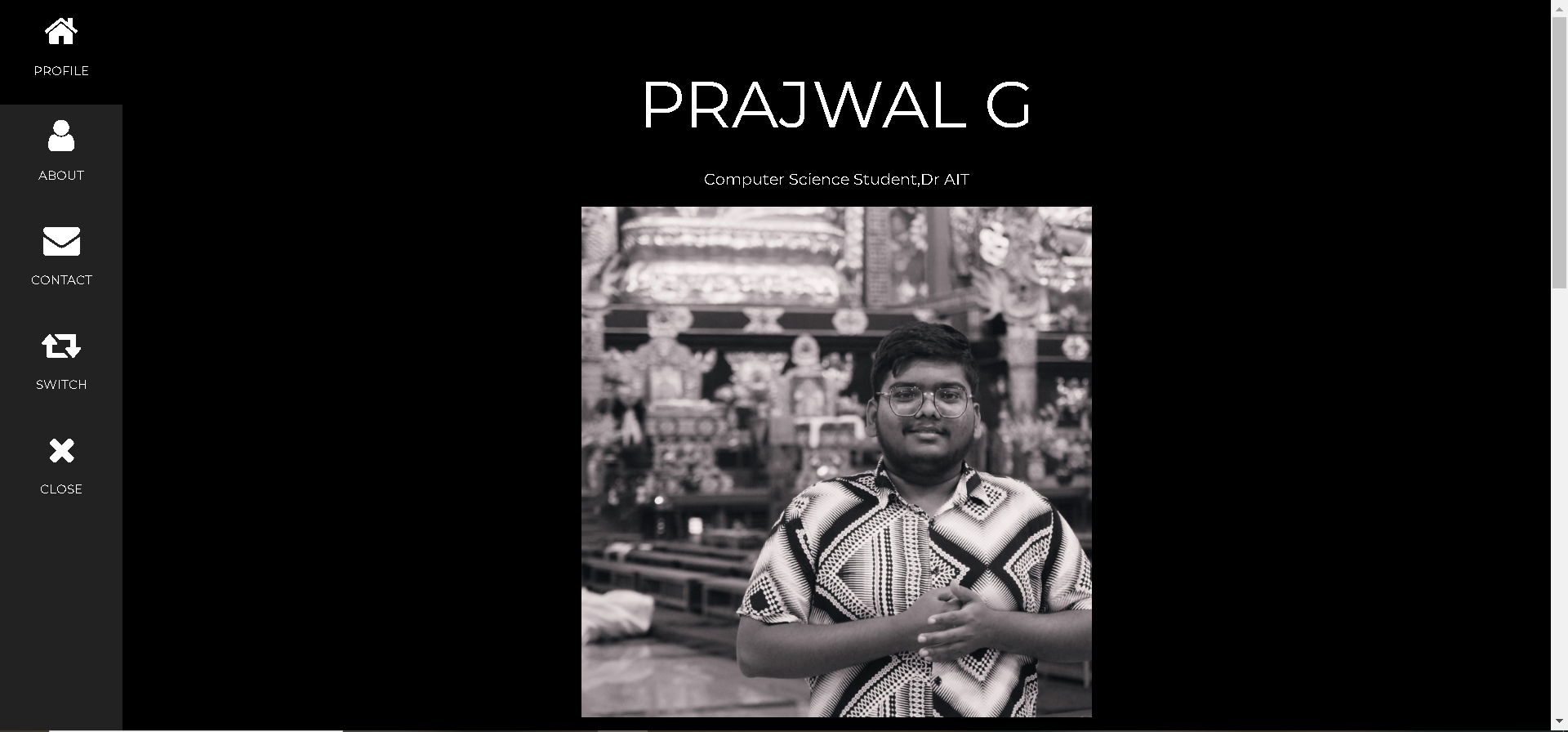
Description automatically generated

Fig 3.4 – Searching Visualizer

A screenshot of a computer

Description automatically generated

Fig 3.5 -Abouts us (Contact Page)



A person in a denim jacket

Description automatically generated

APPLICATION

There are two principal applications of algorithm visualization: research and education. Potential benefits for researchers are based on expectations that algorithm visualization may help uncover some unknown features of algorithms. For example, one researcher used a visualization of the recursive Tower of Hanoi algorithm in which odd- and even-numbered disks were coloured in two different colours. He noticed that two disks of the same colour never came in direct contact during the algorithm’s execution. This observation helped him in developing a better non-recursive version of the classic algorithm. To give another example, Bentley and McIlroy [Ben93] mentioned using an algorithm animation system in their work on improving a library implementation of a leading sorting algorithm.

The application of algorithm visualization to education seeks to help students learning algorithms. The available evidence of its effectiveness is decisively mixed. Although some experiments did register positive learning outcomes, others failed to do so. The increasing body of evidence indicates that creating sophisticated software systems is not going to be enough. In fact, it appears that the level of student involvement with visualization might be more important than specific features of visualization software. In some experiments, low-tech visualizations prepared by students were more effective than passive exposure to sophisticated software systems.

To summarize, although some successes in both research and education have been reported in the literature, they are not as impressive as one might expect. A deeper understanding of human perception of images will be required before the true potential of algorithm visualization is fulfilled.

CONCLUSION

Several searching, path finding and sorting methods are visualized using graphical and animation techniques. It is a useful tool for students and researchers to quickly comprehend the implicit algorithmic sequences. Here, users are welcome to choose an algorithm from a list while the algorithm is visually explained. It has been discovered the learning through animations rather than merely text or oral explanations makes it easier for humans to remember concepts. The method is designed to be easily accessible. Students can easily understand the time and space complexity as well.

Some features implemented include a feedback system enabling the user to contact us if there exist any queries or even suggestions. While learning from the visualizations the learner can input their own data and see the outcome animated. So, they can experiment with various self-generated examples. They can also control the animation speed along with the options of rewinding, fast forwarding or skipping a particular iteration. Some of the general animation controls implemented in the system are - Skip Back/Forward, Step back/forward, Play/Pause Toggle, Animation Speed (in the form of slider) & Change Canvas Size (to change the width / height of the display area).

A drawback of using algorithm visualizations within our subject is the lack of the tool oﬀering required visualizations in a single package with the uniﬁed interface .

FUTURE ENHANCEMENTS

This system is implemented for visualizing some of the sorting, searching, path-finding algorithms. This is a helpful tool for all kinds of students and tutors to easily understand the execution of algorithms. For future enhancement we can include more algorithms of sorting, searching and path-finding.

A visualization tool for visualizing some basic algorithms along with sorting ,search and path finding algorithms and operations associated with them has been presented. This tool provides an easy way to play and learn sorting ,search and path finding algorithm concepts with its user-friendly and self-explanatory interface. In this system, only some commonly used and basic algorithms are implemented like Dijkstra , depth first search , breadth first search , binary search , linear search , various sorting methods etc. Its scope can be extended by implementing more complex algorithms in the software. It can also be categorized for a more systematic interface. Developing and implementing a mechanism for the software package to recognize the user- defined observable algorithms, and leave the implementation to the user is yet another way to extend its current scope, allowing users to use their own observable data structures, thus adding more flexibility to the software.

Some of proposed core-related features are on the list too (like graphically better visualizations, optional changing of algorithm properties), but some of them will probably not be implemented in a near future (like undo/step back in running visualization), as they would require more fundamental changes.

REFERENCES

K. Becker and M. Beacham, "A tool for teaching advanced data structures to computer science students: an overview of the BDP system," Journal of Computing Sciences, vol. 16, no. 2, pp. 65-71, 2001

E. Vrachnos and A. Jimo Yiannis, "Design and evaluation of a web-based dynamic algorithm visualization environment for novices," Procedia Computer Science, vol. 27, pp. 229-239, 2014.

F. E, A. M and S. C, "The role of visualization in computer science education," Computers in the Schools, vol. 29, pp. 95-117, 2012.

D. Hundhausen, S. Douglas and J. Stasko, "A meta study of algorithm visualization effectiveness," Journal of Visual Languages and Computing, vol. 3, no. 3, pp. 259-290, 2002.

G. Robling and T. L. Naps, "A testbed for pedagogical requirements in algorithm visualizations," in Conference on Innovation and Technology in Computer Science Education, New York, USA, 2002.

J. Urquiza-Fuentes and J. A. Velazquez-Iturbide, "A survey of successful evaluations of program visualization and algorithm animation systems," ACM Transactions on Computing Education, vol. 9, no. 2, pp. 1-24, 2009.

Bremananth R, Radhika V Thenmozhi S “Visualizing Sequence of Algorithms for Searching and Sorting” 2009 International Conference on Advances in Recent Technologies in Communication and Computing2009

T. L. Naps, G. Robling, V. Alms Trum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger and J. A. Velazquez-Iturbide, "Exploring the role of visualization and engagement in computer science education," in Proceedings of ITICSE on Innovation and Technology in Computer Science Education, New York, USA, 2002.