Algorithm Visualizer

PROF. ASHA P[1], GURPREET KAUR[2], ANMOLPREET S[3], SHASHANK K[4], AKSHAY KUMAR[5]
[1, 2, 3, 4, 5] *Guru Nanak Dev Engineering College, Bidar*

Abstract- ***Algorithm analysis and design is a great challenge for both computer and information science students. Fear of programming, lack of interest and the abstract nature of programming concepts are main causes of the high dropout and failure rates in introductory programming courses. With an aim to motivate and help students, a number of researchers have proposed various tools. Although it has been reported that some of these tools have a positive impact on acquiring programming skills, the problem still remains essentially unresolved. This project describes "ALGORITHM VISUALIZER", a tool for visualization of algorithms. Algorithm Visualizer is an easy-to-set-up and fully automatic visualization system with step-by-step explanations and comparison of sorting algorithms. Design principles and technical structure of the visualization system as well as its practical implications and educational benefits are presented and discussed.***

I. INTRODUCTION

How do you work out a problem? The problem itself doesn't need to be anything overly complex, such as trying to replace a broken headlight in your car (although nowadays, manufacturers are trying the patience of the community with their increasingly abstract, space-age designs). The point is how to attack the problem. Do you perform research, such as looking through your car's manual for step-by-step instructions, or is your first instinct to find someone who knows how to do it (whether they are right next to you or in an online video)? My instinct is the latter, as I am a visual learner and am adept to picking up concepts by seeing it done, rather than reading about it. For example, when I was learning about algorithms while pursuing my Computer Science degree, I found that seeing the data move to its correct position under the constraints of an algorithm was much easier to follow than tracing the code by hand. That led to the inspiration of this paper, which describes a web-based tool I created that animates how algorithms modify and organize a set of data.

I wanted the animation to be web-based to appeal to a wide spectrum of people using different technology media. This way, the user would not need to worry about installing special software or trying to organize configurations to use the tool. The webpage is coded with HTML5 (Hypertext Markup Language, Version 5), JavaScript, and CSS (Cascading Style Sheets). The physical elements of the webpage (buttons and layout) are coded with very minimal HTML5 code. The next biggest contributor would be the CSS code, which is responsible for the appearance and behavior of the buttons and text. Finally, the rest is devoted to JavaScript, responsible for the histogram generation, movement, algorithm design, and sound. All the buttons refer to designated parts of the JavaScript code to perform the task.

II. PROBLEM STATEMENT

Algorithms are one of a foundation concept of computer science used to prepare students to further important concepts. Traditionally, students learn this concept by memorizing facts of an execution process of a certain predefined algorithm resulting in having a superficial understanding of the underlying process. The students are deprived their opportunities to solve the problems by themselves, and having low development of problem-solving skills, such as Computational thinking. While computer science students have to solve many novel problems, lacking the essential cognitive skills would be a problem since the problem-solving skills are very crucial in computer science education. In order to promote skills, using visualization is a good mediator because it has many pedagogical benefits and promotes active learning and student-centered learning.

III. DESIGN

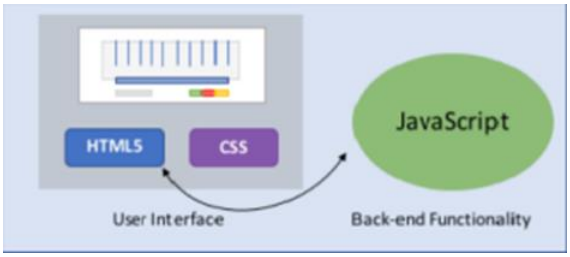
- *The User-Interface*

The user can select any of these algorithms to see the visualization of how that algorithm works. There is no algorithm selected by default, so the user will need to select one before starting the animation.

The user has three options to choose sorting, searching, or path-finding algorithm. Then they have multiple options in the selected algorithm.

- *System Architecture*

The back-end code is comprised of HTML5, CSS, and JavaScript. All three types of code are contained in one .html file and can be run solely from this file. One of the advantages of HTML 5 is that it is not necessary to include different types of web languages in a single file. Therefore, each type could 16 have been separated, making a total of three files (plus the miscellaneous sound and image files). This is good practice for readability and keeping related code together. However, I decided not to separate the code for two reasons: 1) to increase the portability of the project by only needing to worry about one project file instead of three, and 2) where in the project file, the change in coding languages is distinctly marked and therefore does not significantly reduce readability. Also, the ability to put more than one web language in a single file is an example of an RIA (Rich Internet Application).



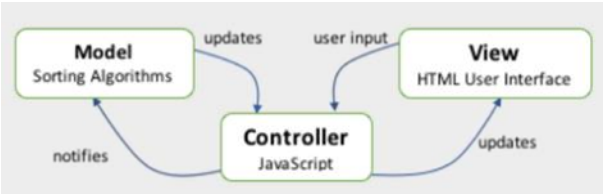
As you can see, there are no major components besides the three coding languages. Most websites have tools or scripts that require a server on the back-end (like PHP), but it is not necessary in this case since JavaScript runs right in the user's browser. HTML5 and CSS are used for the interface. The HTML5 communicates with the JavaScript code and vice versa to launch the 17 appropriate algorithms and update the interface accordingly, as seen with a single, bidirectional arrow. Throughout the project, the code for the HTML5 and CSS did not change much. As the JavaScript was modified from a functional programming focus to a more object-oriented one, the parts of the HTML5 that did change were the function calls for each button. All of the back-end interaction is abstracted to the various buttons for selecting algorithms and running the animation.

IV. IMPLEMENTATION

The implementation of this project is a combination of HTML5 (Hypertext Markup Language 5), JavaScript, and CSS (Cascading Style Sheets). There is only one project file that contains the code and is an HTML file.

The organization of the code follows both object-oriented and functional programming concepts. Originally, the design was almost completely functional, where only three objects were used: one to control the canvas that displayed the animation, another to represent a piece of data, or "bar" object (blue rectangle with dynamically changing height and position), and a final one to represent the positions that each bar moved to, or "pos" objects. Some instance variables and Boolean values were used to keep track of the algorithm selected and when to animate, but this resulted in a heavily integrated mass of function calls that was difficult to upkeep.

One large refractor later, the code now resembles a Model-View-Controller architecture. Although, due to its functional nature, it has many more individualized functions that update the instance variables and Boolean values, thereby directly updating the View and Controller. A simplified diagram of 19 the Model-View-Controller relationship is below in Figure



The first goal of this project visualizes various sorting and searching algorithms, and create a web application to visualize these algorithms. [25] This visualization portion's goal is to create two types of visualization, first is the pathfinding approach and second is sorting visualization approach, In this pathfinding approach's

goal is to create simple javascript logic to visualize searching algorithms like Dijkstra algorithm, A* search algorithm, etc and sorting visualization's goal is to generate random arrays with random size and then create some simple javascript logic to sort these arrays, as a result, we can simply visualize sorting algorithms. This project contains three single-page applications.

In the path finding approach, the authors implement some java script logic to visualize searching algorithms. Here we visualize seven popular searching algorithms - First Dijkstra's Algorithm (weighted): the father of pathfinding algorithms ,guarantees the shortest path, second A* Search (weighted): arguably the best pathfinding algorithm, uses heuristics to guarantee the shortest path much faster than Dijkstra's Algorithm, third Greedy Best-first Search (weighted): a faster, more heuristic-heavy version of A*, fourth one does not guarantee the shortest path, fifth is Swarm Algorithm (weighted) : a mixture of Dijkstra's Algorithm and A*, does not guarantee the shortest-path, sixth is Convergent Swarm Algorithm (weighted): the faster, more heuristic-heavy version of Swarm; does not guarantee the shortest path, seventh is Bidirectional Swarm Algorithm (weighted): Swarm from both sides; does not guarantee the shortest path, eighth is Breadthfirst Search (unweighted): a great algorithm; guarantees the shortest path, ninth is Depthfirst Search (unweighted): a very bad algorithm for pathfinding; does not guarantee the shortest path.

The Controller consists of all visible buttons on the webpage. These buttons are coded in HTML and directly call a JavaScript method when pressed. The CSS is used for stylizing the appearance, layout, color, and effects of the buttons, along with the canvas. The methods manipulate the state of the Model and perform the necessary updates to prepare and update the View for animation.

This animation can be controlled by the buttons: Start, and Stop,. These update the current state of the bar graph by pulling new values from the step array. The View visualizes these updates and the result is the bars changing position based on what algorithm was selected. The Start and Stop buttons only control a timer that either calls the Step button at a predetermined speed (every quarter of a second) or stops the timer, respectively.

V. 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.

CONCLUSION

Through much time and effort, I have successfully created a working web-based animation tool for visualizing the following sorting algorithms: Selection Sort, Bubble Sort, Insertion Sort, and Merge/Insertion Sort and path finding, search algorithm.

Even with its memory overhead, it received overall positive feedback from the students who explored it. I am not surprised that there was not a significant difference in learning the material, which reflects what

I found in my previous research. There remains, however, a strong mindset to research and create animations like these to improve learning in the classroom, which I agree with completely.

Learning how to code a web platform was challenging, and I thank the tutorials on W3Schools.com for getting me there. I had a course where I updated the JavaScript on a webpage, but it was much more concise and did not involve objects and HTML5 for visualizations. The good news is that JavaScript is still one of the most popular web languages, so I am not too worried about another big refactor soon for a language update.

REFERENCES

[1] https://digitalcommons.ric.edu/cgi/viewcontent.cgi?article=1129&context=honors_projects

[2] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.9207&rep=rep1&type=pdf

[3] https://theses.cz/id/3w4s3a/Mykhailo-Klunko-bc-thesis.pdf

[4] https://www.geeksforgeeks.org/fundamentals-of-algorithms/

[5] https://panthema.net/2013/sound-of-sorting/

[6] https://algorithm-visualizer.org/

[7] https://clementmihailescu.github.io/Pathfinding-Visualizer/