Algorithm Visualizer

Ashwani Kumar Singh, Danish Jamal and Pranjal Aggarwal

April 25, 2021

Algorithm Visualizer

Ashwani Kumar Singh
School of Computer Science
Galgotias University
Gr. Noida, India
ashwanicena5@gmail.com

Danish Jamal
School of Computer Science
Galgotias University
Gr. Noida, India
danishjamal.104@gmail.com

Pranjal Aggarwal
School of Computer Science
Galgotias University
Gr. Noida, India
pranjalaggarwal2708@gmail.com

***Abstract*—Algorithm visualization illustrates how algorithms work in a graphical way. It mainly aims to simplify and deepen the understanding of algorithms operation. Within the paper we discuss the possibility of enriching the standard methods of teaching algorithms, with the algorithm visualizations. As a step in this direction, we introduce the Algorithm visualizer platform, present our practical experiences and describe possible future directions, based on our experiences and exploration performed by means of a simple questionnaire**

***Keywords*—algorithm visualization • plugin-based visualization platform • computer science education**

I. Introduction

Algorithms and data structures as an essential part of knowledge in a framework of computer science1 have their stable position in computer science curricula2, since every computer scientist and every professional programmer should have the basic knowledge from the area. With the increasing number of students in Central European's higher education systems in last decades (more concrete numbers and impacts for the case of Slovak one can be found in), introduction
of appropriate methods into the process of their education is also required. Our scope here is the higher education in the field of computer science. So within the paper, we discuss the extension of standard methods of teaching algorithms, using the whiteboard or slides, with the algorithm visualizations. According to they can be used to attract students' attention during the lecture, explain concepts in visual terms, encourage a practical learning process, and facilitate better communication between students and instructors. Interactive algorithm visualizations allow students to experiment and available nowadays, and results are quite encouraging. A systematic meta-study of 24 experimental studies can be found in. Results of empirical study aimed at the determination of factors influencing the effectiveness of algorithm visualization are published in. Another example is the study with the objective to determine learning advantage of the interactive prediction facility provided by the courseware containing algorithm animations and data structure visualizations. Based on above mentioned reasons, results of studies carried, as well as our own experiences and explorations, we consider algorithm visualization important and perspective area of further research and application of its results in nowadays computer science education. Except the algorithm visualization, the term software visualization is also often used within the papers published in last
years. It usually covers both visualization of algorithms and visualization of data structures, but sometimes also another aspects of software (like its development process) are considered, too. Algorithm visualization, as part of software visualization, could be described as "graphical representation of an algorithm or program that dynamically changes as the algorithm runs". An overview of visualization taxonomies, together with an analysis of factors increasing the
effectiveness of software visualization, is summarized in . Even if the beginnings of algorithm visualization date back into 1940's , the greatest development in the area we could observe within the last 20-30 years. Modern approaches to software visualization were brought in the 1980's by the introduction of system BALSA (Brown & Sedgewick, Brown University, USA). Some of contemporary solutions include systems like TRAKLA23, ANIMAL4, JAWAA5 or Algorithms In Action6. Concise overview of development in the area of software visualization we provided in, so it is not our intention to analyse this topic within the paper.

II. Algorithm Visualizer

In addition to the mathematical and empirical analyses of algorithms, there is yet a third way to study algorithms. It is called algorithm visualization and can be defined as the use of images to convey some useful information about algorithms. That information can be a visual illustration of an algorithm's operation, of its per-formance on different kinds of inputs, or of its execution speed versus that of other algorithms for the same problem. To accomplish this goal, an algorithm visualization uses graphic elements points, line segments, two- or three-dimensional bars, and so on to represent some "interesting events" in the algorithm's operation.

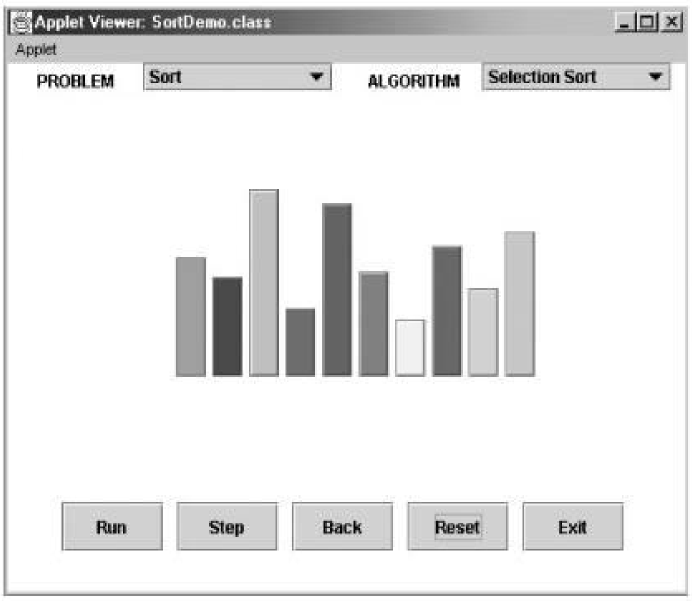
III. Background of Algorithm Visualization

Despite that arrays constitute a fundamental data structure in introductory programming curriculum, only a small amount of research has been directed to the investigation of students' mental models and programming difficulties with arrays [2, 6, 18, 19]. According to a survey of computing educators, loops and arrays are two of the three programming topics of major difficulty for novice students. Du Boulay reported on students' confusion between an array index and its cell; they also have difficulties to deal with arrays that contain indices as array elements. An unpublished authors' investigation in Greek secondary schools, using a sample of 102 students (K-12), confirmed the same misconceptions and learning difficulties; the majority of the students had faulty or incomplete models of the array concept which resulted in misconceptions and serious difficulties in solving simple algorithmic problems which demand the use of array data structure. In general, there are two main categories of visualization systems in CS education: program visualization and algorithm visualization systems. Program Visualization (PV) systems produce direct representations of programming structures and/or program execution phases (e.g., values of variables, internal program structures, method frames, data structures, objects etc.). Jeliot 3 is a well-known PV system which visualizes Java programs; other contemporary systems, like Jype [20], UUhistle and Online Python Tutor, visualize programs in Python. However, the logic behind an algorithm cannot be revealed by just showing how the values of the program variables change. Students need proper graphical representations which fit better to their mental models about the execution of the particular algorithm. Algorithm Visualization (AV) systems aim to cover this need by visualizing abstract concepts and unfolding the underlying logic of the algorithm under study, thus helping students to construct multiple mental models, to interlink construct hierarchies and generalize problem-solving patterns. The terms static and dynamic algorithm visualizations are used in the literature in order to distinguish the degree of interactivity and students' experimentation with the visualization (e.g. abilities to modify both, input data and algorithm code, as well as various representations of the visual objects). A recent survey about algorithm visualization systems can be found in . The first reference to algorithm animation was the famous video entitled 'Sorting Out Sorting' which has been presented by R. Baecker on 1981 at the SIGGRAPH Conference. This 30 min video demonstrated the characteristics and the operations of nine sorting algorithms, using animation and audio comments. Since then, there were several tools developed as a result of research projects on algorithm visualization. The most popular technique for creating an algorithm animation is by annotating the algorithm code with scripting commands producing the visualization. The first system based on a scripting language has been developed by John Stasko and his colleagues and belongs to a wide family of algorithm visualization systems (Tango, Polka, Samba and JSamba). Animations consist of a file containing graphics instructions which correspond to important events of the algorithm under visualization. Another family of systems, like MatrixPro, Trakla2 and Ville , provide "Algorithm Simulation Exercises", where the student has to manually perform a given algorithm, typically by dragging elements to new or target positions or by clicking on buttons to cause a certain function. Ville is a new tool of this family; it supports multiple programming languages includibg C++ and Java. Its built-in editor supports creation of interactive quizzes and tests displayed as pop-up windows. Another novel AV system is JHave which helps AV developers to easily create animated slideshows. Its specific feature is the 'stop-and-think' questions and explanations that can appear at any time during the execution of the animation, thus promoting students' active interaction with the visualization of the algorithm. JHave includes a large collection of algorithm visualizations and has received great educational interest. A recent algorithm animation system is Alvis Live!. It is a program development environment that supports construction and interactive presentation of algorithm visualizations using SALSA scripting language. It includes features that support story boarding. Moreover, Alvis Live! provides an error checking system, which reports error messages while the student develops his own code

A. Static Algorithm Visualization

Algorithm animation, on the other hand, shows a continuous, movie-like presentation of an algorithm's operations. Animation is an arguably more sophisticated option, which, of course, is much more difficult to implement.

Early efforts in the area of algorithm visualization go back to the 1970s. The watershed event happened in 1981 with the appearance of a 30-minute color sound film titled *Sorting Out Sorting*. This algorithm visualization classic was produced at the University of Toronto by Ronald Baecker with the assistance of D. Sherman [Bae81, Bae98]. It contained visualizations of nine well-known sorting algorithms (more than half of them are discussed later in the book) and provided quite a convincing demonstration of their relative speeds.
The success of *Sorting Out Sorting* made sorting algorithms a perennial favorite for algorithm animation. Indeed, the sorting problem lends itself quite naturally to visual presentation via vertical or horizontal bars or sticks of different heights or lengths, which need to be rearranged according to their sizes. This presentation is convenient, however, only for illustrating actions of a typical sorting algorithm on small inputs.



IV. Problem Formulation

The application of algorithm visualization to education seeks to help students learn 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. The commonly example of algorithm visulaiozation are.

V. Required tool

A. P5.js

p5.js is a JavaScript library for creative coding, with a focus on making coding accessible and inclusive for artists, designers, educators, beginners, and anyone else! p5.js is free and open-source because we believe software, and the tools to learn it, should be accessible to everyone.

Drawback: No built-in rendering. If looking for something to build UIs with or similar, one might be disappointed by the lack of any predefined UI element objects and such in p5. js. One has to write all the rendering code for any objects one includes, integrating it appropriately with the loop.

B. Pixi. Js

Pixi.js is a rendering library that will allow you to create rich, interactive graphic experiences, cross-platform applications, and games without having to dive into the WebGL API or grapple with the intricacies of browser and device compatibility. https://www.pixijs.com/

VI. Feasibility Analysis

VII. Merits

1) It mainly aims to simplify and deepen the understanding of algorithms operation. Within the paper we discuss the possibility of enriching the standard methods of teaching algorithms, with the algorithm visualizations.

2) Algorithm visualizer platform, present our practical experiences and describe possible future directions,
based on our experiences and exploration performed by means of a simple questionnaire

VIII. Implementation

Algorithms is the very fundamental about programming which is very important for beginner developers and getting it cleared is really tough for certain people. So this implementation will focus on getting the right thing to large number of people in easy and efficient manner.
And here comes the HTML. Website is the easiest and best way of getting to the most of the audience there is always an option of desktop app or mobile application, but those will limit the reach-ability of the idea hence the idea will be implemented using website. Where we will be having website which will be user friendly demonstrating most of the generic algorithms where user can interact and see the algorithm happening infant of them rather than visualize it in dry run or imagining and getting the theory clear.

This implementation is broadly divided into 2 steps which is decided based on various factors, considering the facts that all these three steps can be implemented parallel without disturbing the progress of other. Making the development process smooth and fast reaching our end goal is also the important factor which can't be ignored. So below are the 2 major division of this implementation.

A. Setting Up The Environment

This step is chosen as the first step because is focuses mainly on the front-end where the focus is more towards UI/UX(User Interface/User Experience) rather than the business login, hence this step includes the steps for getting the right dependencies, building user friendly UI and an integration module which easily fits with the business logic to make the product ready. Detailed review and insight of this step will be further discussed.

B. Designing the visualization logic

Any product is a good product only if it has good business logic or back end because if we design a beautiful UI but their is no data to display then it is useless. Hence this step also plays a great role in the final result. Where the main focus is not user driven instead it focuses on developing the visualization function and logic to make the user see what actually is happening in real time by developing a way to implement visualization in real time of different data structures like array, tree, graph etc. More insight about this step is discussed further.

1) Why P5.js

Before getting straight into implementation its important to *get it clear why P5.js is used in this implementation. P5.js is a c*lient side library which is open source and used for delivering graphical and interactive experiences and this idea also deals with the graphical representation. This is one of the main reason for choosing the library. Another reason for choosing P5.js is that it makes the development process really fast as developing animations and motion can be really tough task and we need to develop our own rendering module hence P5.js act as a great tool to eliminate the rendering part so that the main focus is on visualization logic rather than building the rendering module which can be a mini project in itself. P5.js itself is a great tool for making graphical intensive application really light and fast, so concluding the idea this implementation is build upon the library P5.js.

C. Setting Up The Environment

As previously discussed this step involves the steps for getting the right dependencies, building user friendly UI and an integration module which easily fits with the business logic to make the product ready. Detailed review and insight of this step will be further discussed. So first we get the simple website layout ready using simple html tags and the most basic html page with empty values.

```
<!DOCTYPE html>
```

```
<html lang="">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Algorithm Visualization</title>
  <style>
    body {
      padding: 0;
      margin: 0;
    }
  </style>
</head>
<body>
</body>
</html>
```

Next step is importing the P5.js library which can be easily downloaded from their official website
Which comes with the some basic JS(javascript) files and some documentation to integrate it in any website. To maintain the abstraction within our project structure it is necessary to store the files in separate directory, hence the below project structure works well considering the maintainability and abstraction.

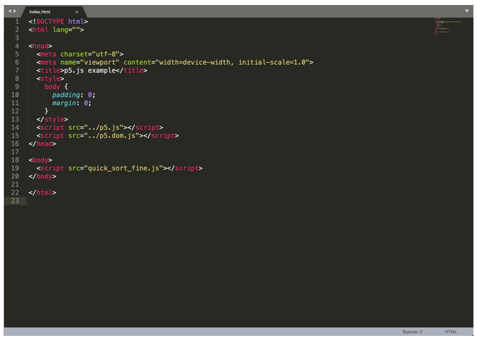
```
——project_dir
         |
         |
        —- p5.js
        —- p5.dom.js
        —- readme.md
        —- addon/
        —- src/
                 |
                 |
                —- src/index.html
                —- src/quick_sort.js
                —- src/bubble_sort.js
```

D. *Structure Explanation*

P5.js and p5.dom.js is the main library files which handles the rendering the sketch file and it needs to be included in main index.html file within the <head> tag.

src/ is the main folder which consist of visualization logic written in java script the detailed structure of sketch is discussed below. All the sketch file resides in the same folder as index.html and it is to be included within <body> tag.

After importing and adding the basic sketch file index would be like:



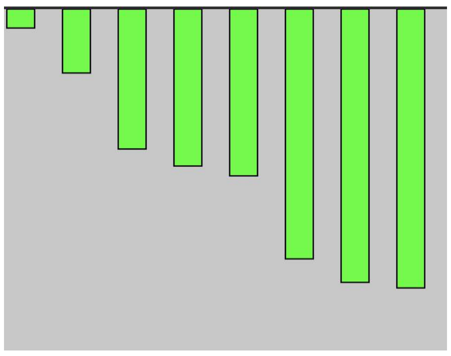
```
<!DOCTYPE html>
<html lang="">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>p5.js example</title>
  <style>
    body {
      padding: 0;
      margin: 0;
    }
  </style>
  <script src="../p5.js"></script>
  <script src="../p5.dom.js"></script>
</head>

<body>
  <script src="quick_sort_fine.js"></script>
</body>

</html>
```

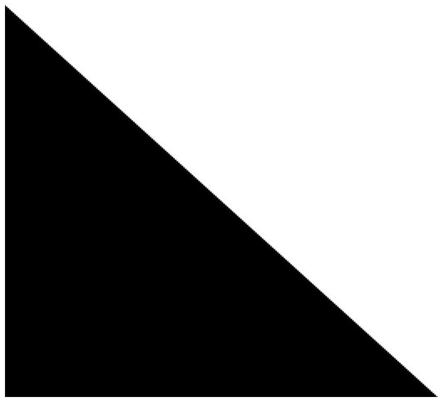
E. *Sketch File*

These are the java script files written in convention of P5.js documentation which basically need two main functions

```
function setup() {

   createCanvas(windowWidth, windowHeight);

}

function draw() {

            // visualisation logic goes here

}
```

The success of Sorting Out Sorting made sorting algorithms a perennial favorite for algorithm animation. Indeed, the sorting problem lends itself quite naturally to visual presentation via vertical or horizontal bars or sticks of different heights or lengths, which need to be rearranged according to their sizes. This presentation is convenient, however, only for illustrating actions of a typical sorting algorithm on small inputs. Hence for dataset the screen is divided in each pixels which is taken as an element in an array and then the array is shuffled using shuffle(array) function. Above logic is used for visualising quick_sort algorithm which makes the. Sorting really fast since sorting each pixel using bubble sort algorithm makes it really lengthy process and slow hence for bubble sort the small and discreet data set is required hence we use bar to represent the bubble_sort visualization the small mock up of what it looks like is show below.



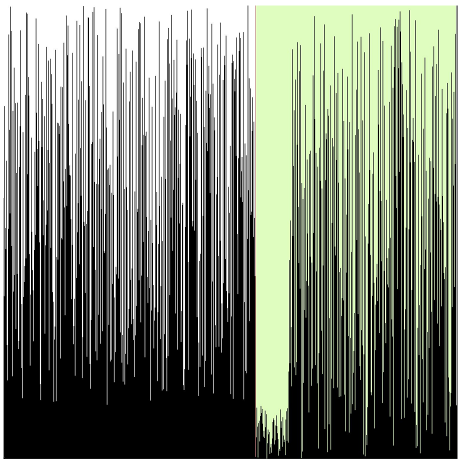


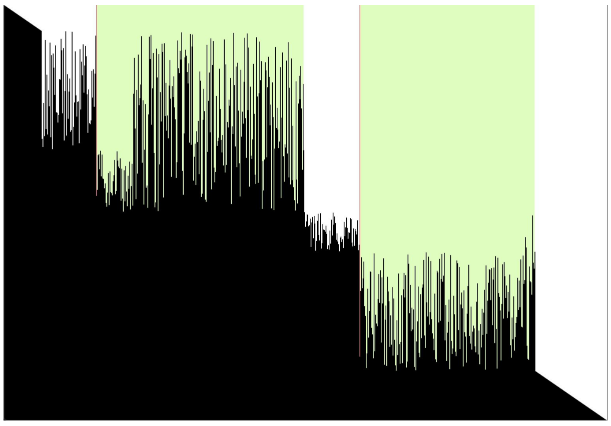
1) Quick Sort Source Code

```
var val;
let i = 0;
let j = 0;
const bw = 20;

let sorted = -1;
preload()
function setup() {
  createCanvas(windowWidth, windowHeight);
  val = [];
  for (let i = 0; i < 30; i++) {
    val[i] = random(1, windowHeight);
  }
  val = shuffle(val);
  console.log(val);
}
function draw() {
  background(200);
  const a = val[j];
  const b = val[j + 1];
  if (a > b) {
    const temp = a;
    val[j] = val[j + 1];
    val[j + 1] = temp;
  }
  if (i < val.length) {
    if (j > val.length - i - 1) {
      j = 0;
      i++;
      sorted = val.length - i;
    } else { j++; }

  } else {
    noLoop();
  }

  for (let i = 0; i < val.length; i++) {
    if (i >= sorted && sorted != -1) {
      fill(0, 255, 0);
    } else {
      fill(255);
    }
    rect(bw * i * 2 + 100, 0, bw, val[i]);
  }

}
```

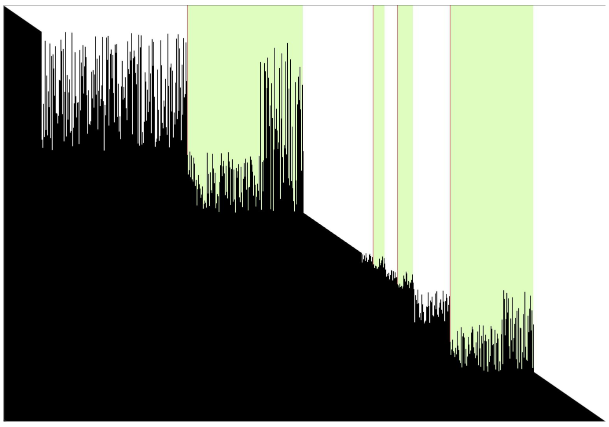
2) Quick Sort Working

Fig(1)

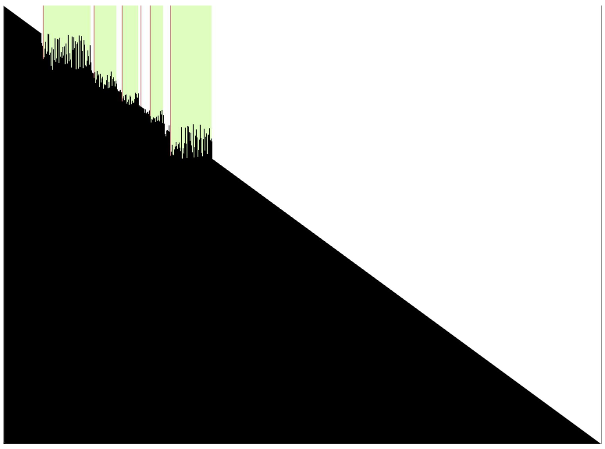




Fig(2)



Fig(3)



Fig(4)

IX. CONCLUSIONS

According to our findings, algorithm visualization can be seen as a valuable supporting tool, used in addition to standard ways of education in the field of computer science. Within the paper we provided an overview of the VizAlgo algorithm visualization platform as well as our practical experiences with the system. We believe (and the results of questionnaire support our belief) it helps to improve the quality of education in the field and contribute to the solution for some of the problems in higher education mentioned at the beginning of the paper.

There are still open issues with using algorithm visualizations. Algorithm visualizations can help understanding the principles, but do not replace the need to implement algorithms by students is a chosen programming language. Another 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. The VizAlgo platform can also be considered as a step in this direction.

Generally, more systematic evaluation of algorithm visualization tools is required, as there is rather informal evidence available that applications of algorithm visualizations are useful [3].

We summarized results of the questionnaire filled in by students in order to support our decisions on further development of the platform, too. Our intentions here include development of new plugin modules from the area of sorting algorithms and more complex data structures. 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. Except the extensions mentioned within the questionnaire, we also consider some other interesting features: dynamic changes in algorithm pseudocode reflected in visualization, different visual views on running algorithm or simultaneous comparison of different algorithm visualizations.

References

[1] K. Mehlhorn, P. Sanders, Algorithms and Data Structures (Springer-Verlag, Berlin Heidelberg, 2008)

[2] J. Genci, Possibilities to Solve Some of the Slovak Higher Education Problems Using Information Technologies, In proceedings of: 10th IEEE International Conference on Emerging eLearning Technologies and Applications, ICETA 2012, Stará Lesná, The High Tatras, Slovakia, November 8-9, 2012

[3] C.D. Hundhausen, S.A. Douglas, J.T. Stasko, A Meta-Study of Algorithm Visualization Effectiveness, J. Visual Lang.
Comput. 13, 259–290, 2002