Immersive Algorithms: Better Visualization with Less Information

Philip Bille
Technical University of Denmark
phbi@dtu.dk

Inge Li Gørtz
Technical University of Denmark
inge@dtu.dk

ABSTRACT

Visualizing algorithms, such as drawings, slideshow presentations, animations, videos, and software tools, is a key concept to enhance and support student learning. A typical visualization of an algorithm show the data and then perform computation on the data. For instance, a standard visualization of a standard binary search on an array shows an array of sorted numbers and then illustrate the action of the algorithm in a step-by-step fashion. However, this approach does not fully capture the computational environment from the perspective of the algorithm. Specifically, the algorithm does not "see" the full sorted array, but only the single position that it accesses during each step of the computation. To fix this discrepancy we introduce the *immersive principle* that states that at any point in time, the displayed information should closely match the information accessed by the algorithm. We give several examples of immersive visualizations of basic algorithms and data structures, discuss methods for implementing it, and briefly evaluate it.

Keywords

Algorithms; Visualization; Immersiveness

1. INTRODUCTION

Visualizing algorithms, with drawings, slideshow presentations, animations, videos, or software tools, is a basic teaching concept used to enhance and support student learning. Software tools that support visualization have been extensively studied over the past few decades [6, 5]

A typical visualization shows the underlying data and a step-by-step computation on top of the data. For instance, a standard visualization of a binary search on a sorted array shows the array and then exeuctes the search on top of the array, while keeping track of the state of the algorithm (a constant number of pointers in the array). However, this visualization does not fully capture the computational environment from the "perspective" of the algorithm. Specifically, the algorithm does not "see" the full sorted array, but only the single position that it accesses during each step of the computation. Indeed, if students see the entire array a somewhat reasonable objection is that they do not need an algorithm to search since they can immediately see if the element is there or not. In this paper, we propose a principle for visualizing algorithms, called *immersive visualization*, that addresses this discrepancy between the perspective of the computational environment of the algorithm and information shown in the visualization. The principle is stated succinctly as follows:

> At any point in time, the displayed information should closely match the information accessed by the algorithm.

The name comes from the idea that students have to "immerse" themselves in the computational perspective of the algorithm. For instance, in a binary search the principle dictates that the visualization should only reveal contents of the array as the algorithm accesses it (see Sec. 2.1).

In this paper, we argue that immersive visualization can enhance student understanding and reasoning about introductory algorithms and data structures, and present several examples of immersive visualizations. We also consider variants of the immersive principle, discuss methods for implementing it in teaching, and briefly evaluate a simple case.

ITiCSE '17 July 03-05, 2017, Bologna, Italy

ACM ISBN 978-1-4503-4704-4/17/07.

DOI: http://dx.doi.org/10.1145/3059009.3072972

2. EXAMPLES

2.1 Searching

Consider a standard binary search in a sorted array A (see e.g. Sedgewick and Wayne [3]). A typical non-immersive visualization of this shows the contents of the A and the pointers `low`, `high`, and `mid` indicating the current range of the search and midpoint of the range, respectively. To enhance immersion we propose to only show the contents at the position accessed by the algorithms (A[`mid`]) and the pointers `low` and `high` (see Fig. 1(a)). This reveals precisely the information used by the algorithm. This also exposes the key property that the array must be sorted for the algorithm to work and thus helps students to reason about the correctness of the computation. Finally, the approach encourages interaction with students (selecting inputs and participation in execution) if the setup supports it (see Sec. 3).

The approach easily extends to other simple algorithms on arrays, e.g., linear search, merging, insertion sort, bubble sort, etc. We leave it as an open problem whether more difficult to visualize algorithms such as merge-sort can be effectively (or should be?) visualized according to the immersive principle.

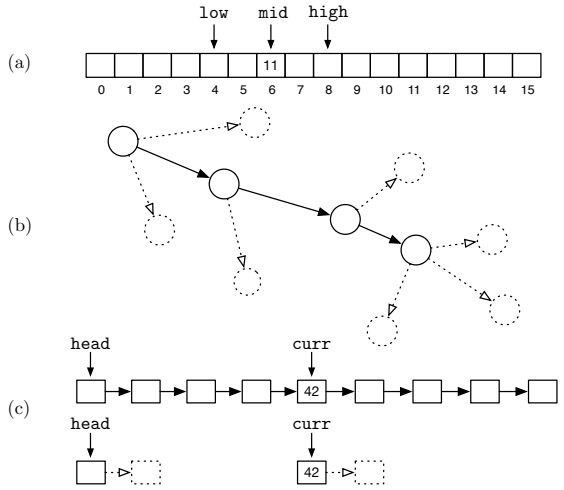

Figure 1: (a) Binary search. (b) Graph search. (c) Partial and fully immersive linked lists.

2.2 Graph Search

Consider a standard textbook depth-first search on a directed graph G [1, 4]. A typical non-immersive visualization shows the full graph and colors on nodes (or marks, labels) as the algorithm explores the graph. We propose an immersive approach, where only the explored nodes and their outgoing edges are visible during search (see Fig. 1(b)). The node coloring can also be updated since this always only affect the explored nodes. Note how this visualization captures the common "maze exploration" analogy for depth-first search [4]. If depth-first search (or it's many variants) is studied later in implicit graph contexts, such as searching game-trees or exploring state-spaces, the immersive visualization already emulates this context.

Breath-first search is slightly more involved, since the algorithm also maintains a queue of nodes during the execution. We propose an immersive visualization that reveals the graphs as it is explored, exactly like depth-first search, but explicitly shows the full queue. The idea is to apply the immersive principle on the graph search part of the algorithm, while viewing the queue as information available to the algorithm.

We leave it as an open problem whether more advanced graph algorithm (shortest-paths, minimum spanning trees, maximum-flow) can be visualized effectively using the immersive principle.

2.3 Linked Lists

Consider a singly linked linked list data structure. A standard non-immersive visualization depicts the linked list as a sequence of nodes each containing data and a pointer to the next element, and a single pointer `head` to access the front the list. We propose two different immersive visualizations depending on learning objectives, students level, and algorithmic focus. The *fully immersive* approach reveals only the node currently accessed by the algorithm and the *partially immersive* approach reveals all nodes and pointers but hides the data stored in each node (see Fig. 1(c)). The fully immersive approach is useful for both teaching basic concepts of pointers, and for teaching advanced manipulation of linked-lists, e.g., constant space reversing or Floyd's cycle-finding algorithm [2]). The partially immersive approach is useful for algorithms that primarily use the data stored in the nodes, e.g., searching the list.

The above approach easily extends to other simple pointer-based data structures or abstract data structures, such as doubly linked list, binary search trees, heaps, stack, queue, tries, etc. For instance, we suggest that the partially immersive approach can be effective in teaching the basic ordering properties of binary search trees and heaps.

3. VISUALIZATION METHODS

We can produce immersive visualization with several methods. Most visualizations made with presentation software (e.g., PowerPoint, Keynote, etc.) can easily be adapted to the immersion principle by hiding and revealing information in each step of the visualization. For classic black-board lectures with drawings we have had significant success with applying immersion in a few cases. For instance, to visualize binary search as described in Sec. 2 we have used cards with numbers attached to a black-board using magnets such that flipping cards reveals or hides the number. This allows for a very simple setup that easily supports interaction with students. We can also imagine simple pointer-based data structures visualized using small cardboard boxes containing hidden pieces of paper (the data) and connected by of strings (the pointers). Numerous software tools for visualizing algorithm have been studied. We are not aware of any software tool that uses the immersive principle. A promising direction for future research is to adapt some of the tools to use immersion.

4. EVALUATION

We conducted a short survey on the interactive binary search visualization on a black-board as described in Sec. 3 in our introductory algorithms courses at the Technical University of Denmark in 2017. Out the 385 students, 215 students responded ($\approx 56\%$). For the question "The visualization helps me understand which data the algorithm accesses" 190 students agreed ($\approx 88\%$), (124 strongly ($\approx 58\%$)). For the question "The visualization helps me understand how the algorithm works" 201 ($\approx 93\%$) students agreed, (134 strongly ($\approx 62\%$)).

5. REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms, 3rd edition, 2009.

[2] D. E. Knuth. *The Art of Computer Programming, Vol 2: Seminumerical Algorithms*. Addison Wesley, 1969.

[3] R. Sedgewick and K. Wayne. *Introduction to programming in Java: an interdisciplinary approach.* Addison-Wesley Publishing Company, 2007.

[4] R. Sedgewick and K. Wayne. Algorithms, 4th edition, 2011.

[5] 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. *Trans. Comput. Educ.*, 10(3):9, 2010.

[6] J. Urquiza-Fuentes and J. A. Velázquez-Iturbide. A survey of successful evaluations of program visualization and algorithm animation systems. *Trans. Comput. Educ.*, 9(2):9:1–9:21, 2009.