Visualizing Geometric Algorithms with WAVE: System Demonstration*
(Short Abstract)

Camil Demetrescu\textsuperscript{\textsection}  Emilio Di Giacomo\textsuperscript{\textsection}  Irene Finocchi\textsuperscript{\textsection}  Giuseppe Liotta\textsuperscript{\textsection}

demetres@dis.uniroma1.it wave@dis.uniroma1.it finocchi@dsi.uniroma1.it liotta@diei.unipg.it

The visual nature of geometric applications makes them a natural area for designing systems which describe relevant aspects of the algorithm behaviour by using animation. Namely, the animation of geometric algorithms is mentioned among the strategic research directions in computational geometry by Tamassia et al. \cite{12} and increasing attention has been put toward designing and developing algorithm visualization facilities for computational geometry (see, e.g. \cite{1, 2, 3, 7, 8, 9, 11}). An algorithm visualization facility receives as input the implementation of an algorithm, an instance of the input of the algorithm, and a specification of the visualization of the algorithm, that is a mapping from the set of algorithms data to a set of geometric shapes that will be displayed; the output is a visual trace of the algorithm execution, i.e. a sequence of moving images that describe the evolution of the data at run-time.

The potentialities offered by World Wide Web for easy communication, education, and distance learning motivates the study of algorithm animation over the Internet. In this context, a common approach to algorithm visualization consists of exploiting the graphic capabilities offered by the Java language without making use of any algorithm visualization facility. This approach has several disadvantages: It is time consuming, requires algorithms to be written in Java, guarantees little protection of the code, and does not separate the task of coding the algorithm from the task of designing its visualization.

Recently, a new approach to algorithm visualization on the Web, called publication-driven approach, that aims at overcoming the aforementioned problems has been presented in \cite{4}. The visualization is specified with respect to an intermediate set of public data structures that are a coherent copy of a subset of the program\textquotesingle s data structures. In the following, we call this intermediate set of data structures public blackboard. With the publication-driven approach, an algorithm visualization is realized by performing two steps: (1) A set of program\textquotesingle s variables, is chosen for publication on the public blackboard. (2) Visualization handlers are attached to the variables of the public blackboard and are executed whenever the variables themselves change. Modifications to the variables in the public blackboard, due to the remote algorithm\textquotesingle s execution, trigger the running of the corresponding handlers on the client\textquotesingle s side. The publication-driven approach allows to clearly separate the task of the Algorithm Developer (i.e. the task of coding an algorithm) from the task of the Algorithm Visualizer (i.e. the task of designing an animation of an algorithm). An ongoing project for the realization of an algorithm visualization facility called WAVE and based on the publication-driven approach is being carried on. After about one year since the project started, a first prototype of WAVE is now available at URL http://www.dis.uniroma1.it/~wave. Aim of the present paper is to briefly describe the current prototype of WAVE.

WAVE is a client/server architecture for algorithm visualization over the Web: End-users can access WAVE through any Java enabled Web browser; in order to limit the communication load on the Internet and to enhance protection of the algorithm implementation, the visualization is displayed by an applet that runs locally on the end-user machine, while algorithms are executed on remote Algorithm Servers maintained by the different Algorithm Developers. The WAVE Server connects the End-user\textquotesingle s applet with the Algorithm Server on which the algorithm to be visualized is executed. The Publisher Server offers to Algorithm Developers a set of functionalities for selecting public variables and enriching the source code with WAVE comments that are used to update the status of the data structures in the public blackboard.

When an Algorithm Developer writes a program that she wants to be visualized, she interacts with WAVE by first downloading the WAVE Front-end which allows to select public variables. It follows the Publication
Phase, which consists of two steps:

**Analysis Phase:** The WAVE Front-end sends the developer code to the Publisher Server, which returns a list of the variables declared in the code. The Developer selects a subset of these variables by means of a simple and friendly interface. The selected variables are maintained in the public blackboard during the algorithm visualization.

**Annotation Phase:** The Publisher Server enriches the developer code by annotating it with a comment at each relevant event (i.e., declarations, assignments, undclarations, etc.) in the life cycle of the selected variables. The enriched code is now ready to be compiled and made available on the Algorithm Server at the Developer’s side.

The visualization is designed by the Algorithm Visualizer who defines a set of graphic visualization routines written in a suitable WAVE language. Visualization routines (also called visualization handlers) are associated with the relevant events of the public variables’ life cycle: any time that a relevant event happens during the execution of the annotated code, a WAVE comment is executed which updates the public variable and triggers the graphic visualization routine.

The End-user GUI is an applet that displays a canvas where the End-user can define the input for the algorithm to be animated. It is possible to save the input, to zoom in and out, and to replay the visualization back and forth at different speeds. Furthermore, the End-user is given the capability of customizing the visualization by changing some of the visualization routines with a friendly user interface.

In order to experiment the effectiveness of the publication-driven approach, we are developing algorithms and designing their visualizations with WAVE. Among the visualized algorithms, we currently have:

- Algorithms for computing dominance drawings of graphs, which display the transitive closure by means of the geometric dominance relation among points in the plane [5, 6]: We implement the visualization of an algorithm where each vertex has both the $x$ and $y$ coordinate greater than those of its dominated vertices, and the visualization of an algorithm where each vertex has both the $x$ and $y$ coordinate greater than or equal to those of its dominated vertices.
- Algorithms for computing several types of proximity graphs, which display closeness relations between pairs of points in a 2D point set (see, e.g., [10]). We visualize algorithms that compute Gabriel graphs, relative neighborhood graphs, and $\beta$-skeleton graphs.
- Some basic algorithms on graphs, such as Depth First Search and st-numbering. The visualization of these algorithms has been an interesting test suite for WAVE on its own, and also an essential component for the visualization of the graph drawing algorithms.

In the system demonstration we show both the Publication Phase for graph drawing and geometric code, a set of visualization handlers designed for them, and several examples of interaction between an End-user and WAVE.

**References**


