GASP-II: A Geometric Algorithm Animation System for an Electronic Classroom

Maria Shneerson
Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot, Israel, 76100

Ayellet Tal
Department of Electrical Engineering
Technion – Israel Institute of Technology
Haifa, Israel, 32000

Abstract

This paper describes GASP-II, an algorithm animation system for geometric computing, which is particularly suitable for a distributed electronic classroom. The system allows the creation, presentation and interactive exploration of 3D geometric algorithms. GASP-II can also be used as a visual debugger for geometric computing.

1 Introduction

Geometric algorithms can be highly complex and difficult to grasp. Algorithm animations can provide the best way to help explain algorithms and to visually debug code. We present in this paper a system, GASP-II, that focuses on algorithms with a geometric flavor, which is accessible to even naive users.

Most algorithm animation systems (e.g., Balsa [5], Balsa-II [2], Tango [9], Zeus [3], CAT [4]) are designed for fundamental algorithms. Their support for 3D geometric algorithms is limited. There exist a few special purpose systems for geometric computing that come to ease the unusual difficulty of the implementation cycle (e.g., [7], [8], [6]). The Computational Geometry Workbench [6] and the XYZ GeoBench [8] support algorithm animations, but this is not their chief goal, and thus creating an animation is still not trivial. Mocha's [1] major objective is to provide algorithm animations over the World Wide Web. It focuses on 2D algorithms. GASP [10] goes beyond these systems by providing a rich set of 3D visualization and animation tools.

While GASP is mainly oriented towards debugging, GASP-II focuses on education. It allows the presentation and exploration of 3D geometric algorithms in a distributed electronic classroom. With GASP-II, views of the same running algorithm may reside on different machines, that can be distributed in various geographic locations. A fundamental difference between GASP-II and most other electronic classroom systems is that GASP-II is interactive. While in most other systems students are completely passive and the system simply displays the same X window on multiple machines, GASP-II allows the students to actively participate in the process of learning by interacting with the algorithm and collaborating with the instructor.

Each student can choose a different *style* for the animation which is suitable to the student's taste. This is done through a comfortable user interface without any need to write or modify any code. During a lecture, each student views a different animation of the same running algorithm. GASP-II also provides every student with a control panel for starting, pausing, stopping, rewinding and controlling the speed of the animation. It is thus possible to rewind and rerun the difficult parts of the algorithm until they are fully understood.

To support debugging, GASP-II provides a rich library of high-level operations to create an animation. During debugging, it is possible to run three communicating processes, GASP-II, a standard debugger, and the user's program simultaneously. The single stepping capability of the debugger combined with the ability to move both forwards and backwards in GASP-II makes it easy to spot bugs visually that would be nearly impossible to find with a standard ASCII debugger.

GASP-II is designed so that each machine is in charge of the graphical work displayed on it. It allows each student to have a full control over the animation. In addition, potential network traffic bottlenecks are avoided since only a relatively small amount of short messages are sent over the network. Moreover, in contrast to the X approach, the least powerful machine does not determine the rate of the whole class.

The next section describes how GASP-II can be used in an electronic classroom. In section 3 we discuss the programmer interface. Section 4 discusses implementation issues. We conclude in section 5.

2 End-User Perspective

GASP-II supports two modes of studying – *self-study* and *guided-study*. In the self-study mode, the lecture becomes similar to studying a chapter from an interactive electronic textbook. In the guided-study mode, the students must follow the instructor throughout the lecture. It is up to the instructor to decide when, during the lecture, a self-study mode is appropriate and when the students should be guided by the instructor. In the latter case, GASP-II ensure that all the views are synchronized with the instructor's view.

An unlimited number of students can be viewing the same algorithm, and they need not be at the same geographical location. Fig. 1 shows three snapshots of screens taken at

[0]This work has been supported in part by the Israeli Ministry of Science, Eshkol Grant 0554-2-96.

Computational Geometry '97 Nice France

the same time. Each screen consists of three windows: the *Control Panel* through which the animation is controlled, the *Algorithm Window* in which the animation runs, and a *Text Window* in which the algorithm is explained.

The Control Panel (on the top left of Fig. 1(a)) contains *general controls* and *study-mode controls*. Exploring the animation is performed through the VCR-like general controls. The viewer can run the algorithm at varying speeds and "rewind" the algorithm in order to observe the confusing parts of the algorithm multiple times - an invaluable tool both for education and for debug.

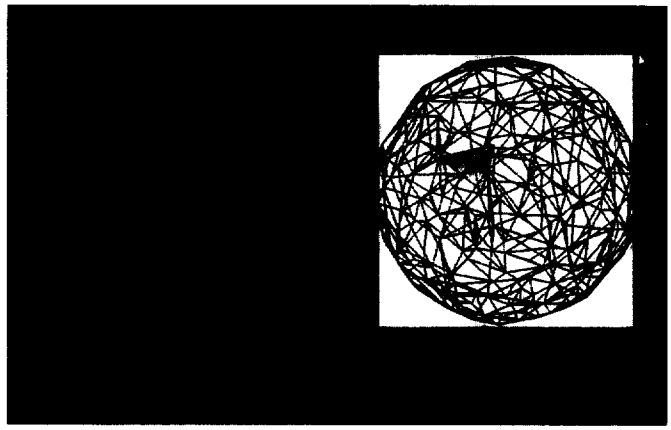
The instructor's control panel differs from the student's control panel in the study-mode controls. The instructor can choose between a self-study mode and a guided-study mode. In the latter case, the general controls at the students' screens are disabled, and the students must follow the instructor in the course of the lecture. GASP-II supports an electronic discussion between the instructor and the students. An electronic discussion is more than a text-exchange, since it is accompanied by a 3D animation of the algorithm. The instructor controls the electronic discussion through this panel.

The student's control panel supports two levels of synchronization. With *instructor unit* synchronization, the animation viewed by the student shows the same phase of the algorithm as the instructor's. With *instructor home position* synchronization, the current camera positions on both screens are the same. The student can then observe the scene in the exact same way as the instructor.

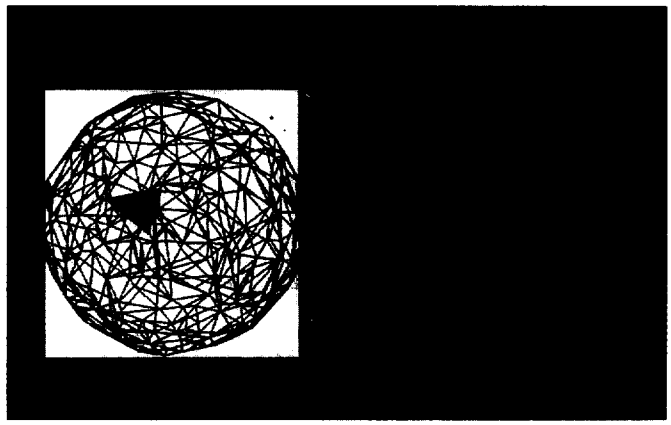
The Algorithm Window (on the right hand side of Fig. 1(a)) is where the algorithm is observed. Different views of the same algorithm animation are shown in Fig. 1(a) - 1(b). In particular, different colors and transparency values were chosen by the first student. In Fig. 1(c) a different phase of the algorithm is being animated. The student rewinded the algorithm to a previous phase.

End-users can modify the default animation created by the system in order to accommodate their personal taste and to enhance their understanding. The preferred animation is selected via a style panel, without any need to write code. An important advantage of GASP-II over other animation systems is that different graphical views should not be programmed beforehand. GASP-II can automatically generate many animations for the same algorithm.

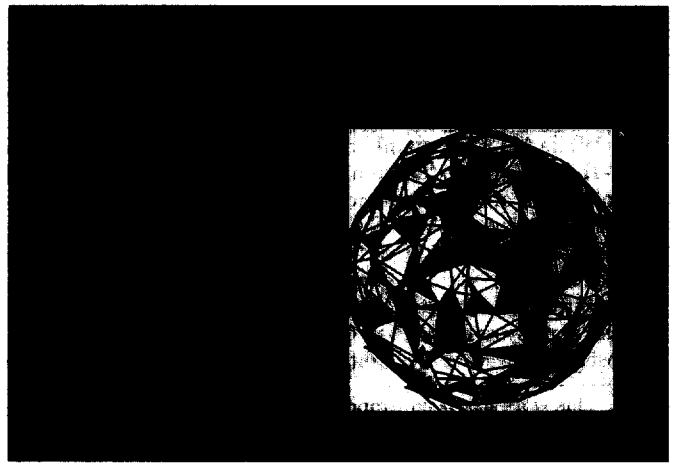
The Text Window (on the bottom left of Fig. 1(a)) is used to elucidate the events and to direct the viewer's attention to specific details. Every logical unit of the algorithm is associated with a piece of text which explains the events occurring during this phase.



(a) The Instructor's Screen



(b) A Student's Screen



(c) Another Student's Screen

Figure 1: Snapshots of Screens

3 Programmer Perspective

The main principle behind the model of GASP-II is that programmers should not be concerned with the presentation aspects of the animation and can choose to be isolated from any decisions of a graphical nature. Instead, programmers should concentrate solely on the contents of the animation. How the algorithm is visualized is left for the system to automatically determine. GASP-II is capable of doing so because it has knowledge about the geometric domain and the proper way to visualize it.

Our framework follows the GASP approach in which the the important logical phases of the algorithm, its *atomic units*, should be identified. Each atomic unit groups together the geometric operations that belong to the same logical phase of the algorithm, and GASP-II animates these

operations as a single unit. Each atomic unit consists of one or more calls to functions in the GASP-II library.

This library provides a rich set of functions which enable the programmer to write animations with minimal effort. The library contains functions which animate the creation, modification, and removal of geometric objects in 2D as well as in 3D. Each such function generates a smooth animation which is appropriate for the required operation.

The application transfers to GASP-II only parameters which are relevant to the geometry (e.g., the vertices and the faces of a polyhedron). No parameters of a graphical nature (e.g., colors, number of frames, the rate of the animation, the way each operation is animated) are required. GASP-II determines all these parameters. This is the major strength of the system, that contains sufficient knowledge regarding the appropriate visualization being generated.

Should the users wish to modify and extend various visualization aspects of the animation, they can change a few parameters in the style panel. Even if the style is modified, the animation is still generated automatically by the system, only a different animation will be generated. The style affects the animation, not the implementation!

4 Implementation

GASP-II's architecture is illustrated in Fig. 2. Two processes, which communicate with each other through messages, are running on the instructor's machine. Process 1 consists of the application code and the GASP-II library. Process 2 is responsible for executing the animation on the instructor's machine and for sending instructions regarding the animation to the students' machines. The process running on each of the students machines is in charge of executing the animation and handling the student's input.

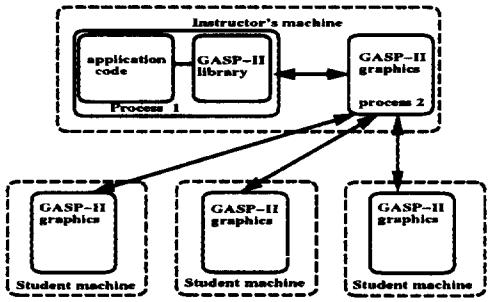


Figure 2: GASP-II's Architecture

The application initiates a call to a GASP-II function. Process 1 prepares messages containing the relevant information regarding this operation, and sends them to Process 2. Utilizing a hand-shaking approach, Process 1 then waits for a reply. Upon receiving the messages, Process 2 consults its style information and executes the animation. It then broadcasts the messages to the students' machines, and immediately sends an acknowledgment to Process 1 without waiting for a response. Process 1 can then return to the application's code. The application now proceeds executing the algorithm. Meanwhile, on each of the students' machines, an animation is executed.

The hand-shaking approach used between the processes on the instructor's machine, comes to facilitate debugging, when the system is used by a single user. It allows the user to visualize the scene at the time when the calls occur. Since rendering is done within an event mainloop, it is otherwise difficult to return to the application after each call.

In a distributed environment, the above architecture has a few important advantages. First, the graphical work is distributed among all participants in the classroom. Second, this architecture allows the students to choose their own styles for a specific animation and view completely different animation for the same algorithm. This is so because the messages received by the students' machines do not specify how the animation should be carried out, and thus each machine is free to determine the appropriate style. Third, since the information sent is very high level, the network traffic is very low. Fourth, if synchronization is required, it can be easily handled. Fifth, if a failure occurs at a student machine, it does not affect the other machines. Sixth, the system fully protects the algorithm code which is never sent over the network. Finally, since each machine can specify in the style panel parameters that fit this machine, machines on the lower-end do not slow down the whole class.

5 Conclusion

GASP-II is an algorithm animation system for geometric computing. It is particularly designed for use in an electronic classroom, yet is also suitable as a debugging tool. With GASP-II, programmers can produce "quick-and-dirty" animations by annotating their code with very short excerpts of C code. Users can experiment with various animations by changing the system's default style through a style panel, without ever modifying or compiling the code. Students can experiment with a finished animation and collaborate with other users.

We believe that animations can be used as a central part of teaching computational geometry, both for demonstrating algorithms, and for accompanying programming assignments. Finally, many possibilities exist in making an electronic book out of GASP-II.

References

[1] J.E. Baker, I.F. Cruz, G. Liotta and R.Tamassia. Algorithm Animation over the World Wide Web. *Proc. Int. Workshop on Advanced Visual Interfaces (AVI '96)*, pages 203-212, 1996.

[2] M.H. Brown. *Algorithm Animation.* MIT Press, 1988.

[3] M.H. Brown. Zeus: A system for algorithm animation and multi-view editing. *Computer Graphics*, 18(3):177–186, May 1992.

[4] M.H. Brown and M.A. Najork. Collaborative Active Textbooks: A Web-Based Algorithm Animation System for an Electronic Classroom. *SRC Research Report 142*, May 1996.

[5] M.H. Brown and R. Sedgewick. Techniques for algorithm animation. *IEEE Software*, 2(1):28–39, Jan 1985.

[6] P. Epstein, J. Kavanagh, A. Knight, J. May, T. Nguyen, and J.-R. Sack. A workbench for computational geometry. *Algorithmica*, 11(4):404–428, April 1994.

[7] K. Mehlhorn and S. Naher. Leda: a library of efficient data types and algorithms. *Technical Report A 04/89, FB10*, Universitat des Saarlandes, Saarbruken, 1989.

[8] P. Schorn. *Robust Algorithms in a Program Library for Geometric Computation.* PhD thesis, Informatik-dissertationen eth zurich, 1992.

[9] J. Stasko. Tango: A framework and system for algorithm animation. *IEEE Computer*, September 1990.

[10] A. Tal and D.P. Dobkin. Visualization of Geometric Algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 1:194–204, 1995.