A Workbench for Computational Geometry

P. Epstein, 2 J. Kavanagh, 3 A. Knight, 4 J. May, 2 T. Nguyen, 5 and J.-R. Sack 6

Abstract. We describe the design and implementation of a workbench for computational geometry. We discuss issues arising from this implementation, including comparisons of different algorithms for constant factors, code size, and ease of implementation. The workbench is not just a library of computational geometry algorithms and data structures, but is designed as a geometrical programming environment, providing tools for: creating, editing, and manipulating geometric objects; demonstrating and animating geometric algorithms; and, most importantly, for implementing and maintaining complex geometric algorithms.

Key Words. Algorithms, Computational geometry, Implementation.

1. Introduction. This paper describes a geometric computing environment we have developed, and which we refer to as the workbench for computational geometry (or the workbench for brevity). We summarize the design concepts and some major features, illustrate the workbench in operation (see, e.g., Figure 1), and discuss issues arising from this implementation-oriented work; we also give empirical comparisons of implemented algorithms performed using the workbench.

This section motivates the work and introduces the workbench at a high level. The remainder of the paper covers certain aspects of the system in greater detail. The design goals of the project and the approaches taken to realizing them are listed in Section 2. The application of object-oriented design is described in Section 3. Possible applications of the workbench are listed in Section 4. Empirical performance studies of algorithms and a summary of the results of these studies are given in Section 5. In Section 6 the system operation is illustrated with a series of screen snapshots, and Section 7 concludes with a summary and mentions ideas for future work.

1.1. Motivation. During the past decade, algorithms have been developed for solving a wide spectrum of problems arising in computational geometry. Research activities have focused mainly on the design and analysis of geometric algorithms.
However, many of these algorithms have never been implemented. Those that have been implemented have often been treated in isolation.

As a consequence, much of this large body of knowledge remains inaccessible to application-oriented researchers from within the computational geometry community and from areas such as computer graphics, robotics, image processing, and pattern recognition which have been the source of many geometric problems.

Thus a "geometric computing environment" (as termed in [F1]) which can provide a unified framework and a suite of tools to reduce difficulties in the implementation, application, and evaluation of geometric algorithms is needed. Furthermore, within the computational geometry community an increased interest can be observed for more direct comparisons of algorithms in terms not only of worst-case complexity, but of execution time and space (including constant factors), ease of implementation, robustness, handling of special cases, and average-case performance. Information on these characteristics is important in assessing the quality and usefulness of an algorithm, particularly to the implementation-oriented researcher.

The increased interest in the study of implementation issues for geometric algorithms has resulted in at least two other projects: The LEDA project [MN] by Mehlhorn and Näher, and the XYZ GeoBench by Nievergelt and Schorn [Sc]. LEDA is a library with implementations of a large number of graph algorithms and well-designed data types moving toward the implementation of computational geometry algorithms. The XYZ GeoBench is similar to our project in that it provides a user interface as well as a library, but its main focus is on robust implementation of fundamental algorithms. Both of these systems are implemented in an object-oriented style.

1.2. High-Level Description. The algorithmic portion of the workbench may be viewed as consisting of four layers (depicted in Figure 2). The base layer contains the primitive operations; these are the fundamental components upon which the

Fig. 1. (a) A triangulated 100 vertex polygon, (b) The shortest-path tree from a query point, generated by the workbench.