Available online at www.sciencedirect.com
ScienceDirect
Computational Geometry 38 (2007) 1–3
Computational Geometry
Theory and Applications
www.elsevier.com/locate/comgeo

Editorial

A decade of CGAL

The Computational Geometry Algorithms Library CGAL is the largest and most influential collection of algorithms and data structures for geometric computing that is available today. It started off in 1996 as a European research project with a small number of partners. Over the last ten years, it has grown tremendously, now being a major open source project with a lot of infrastructure (see www.cgal.org). Contributors come from the original partner sites, from sites involved in subsequent research projects, but increasingly also from many other sites worldwide.

In these ten years, CGAL has fostered a new computational geometry research culture, and it has substantially shortened the time that it takes to turn academic results into industrial-strength software. For the editors of this special issue (both being with CGAL from day one), it is a great pleasure to see how the computational geometry community has widely adopted CGAL as a platform that helps in teaching, testing ideas, writing research papers, and building complex software. The papers of this special issue are a living proof for this adoption. The earned reward for the community is that people in many application domains can now successfully use the clever and ingenious techniques developed by computational geometers.

This was not always the case. Back in the early nineties, the field of computational geometry was going through an adolescence crisis. Since the early eighties—when the field had come into existence as a distinguished subfield of algorithms design—many of the classical problems had been resolved, with much enthusiasm and many brilliant new ideas. This resulted in a computational geometry theory, a necessary body of fundamental results.

In building this theory, its potential relevance in applications was never in doubt, but actual claims of applicability were scarcely substantiated. Computational geometry algorithms were designed for an idealized model of computation (the real RAM), and they often disregarded certain special situations as "degenerate" and irrelevant for the asymptotic results. On the one hand, only these abstractions had enabled the field to advance so quickly, but on the other hand, they now proved to be a serious obstacle toward practical implementations: computers simply did not work with infinite precision numbers, and real world problem instances were full of exactly those special situations that one never saw in testing with randomly generated synthetic data.

In order not to become irrelevant in a world of vastly growing application challenges, the computational geometry community had to reconsider the basis of its theory. Within the European research project ALCOM, the software library LEDA was developed. LEDA also contained a number of geometric algorithms, designed for actual computers, and prepared to deal with all the situations that might arise. Also, a first version of the CGAL kernel was developed by then. The time was ripe for a fully-fledged library of computational geometry algorithms, and the European project CGAL took up the challenge. Quoting from the project proposal: "The goal of the CGAL project is to make the large body of geometric algorithms, developed within the field of computational geometry, available for industrial applications... Even though for most problems we want to solve in the library, correct and efficient algorithms have been published, these cannot immediately be turned into robust and efficient code."

The CGAL project, funded by the European Union, started in 1996. The fact that the project was on the right track was confirmed when the Computational Geometry Impact Task Force published its report *Application Challenges to Computational Geometry* in the same year. This task force, installed as a reaction to the computational geometry crisis, issued concrete recommendations for the future of the field. Here are two quotes from the report: "An on-line library of geometric code available through the Internet would be a useful starting point." and "Building novel geometric

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2007.02.001

software that transforms the practice of geometric computing should be considered on par with proving a theorem that changes the mathematical landscape of the field."

The CGAL project was succeeded by other European projects (GALIA, ECG, ACS), all with the goal of strengthening application-oriented computational geometry in general and CGAL in particular. During all these projects (and until the present day), CGAL has been the major platform for the development of new software.

The fact that CGAL has become what it is now, is also a consequence of design decisions in the early phase that—in retrospect—turned out to be very fortunate. For example, CGAL was already committed to generic programming when this paradigm was new and only insufficiently supported by compilers. In fact, CGAL contributes to the state of the art of generic programming. The support of the exact geometric computation paradigm is one of the hallmarks of CGAL. These aspects make CGAL an ideal platform for implementing geometric algorithms and for experimenting with different number types, data structures, and algorithms.

Today, CGAL is a software platform that is well-integrated with computational geometry research. CGAL itself works according to good scientific practice: software submissions are reviewed under the guidance of the Editorial Board, and regular developer meetings with technical talks ensure that the project makes progress as a whole. CGAL is also an open source project exploiting extensive test suites and version management, and having elaborate user and developer manuals. And CGAL has also become a well-established commercial software platform, with customers worldwide, and in many application domains. The *Geometry Factory* sells software and support around CGAL, and it facilitates the transfer of academic software into industrial-strength code.

The goal of this special issue is to present original research that describes contributions to CGAL, or applications that critically use CGAL. The selected papers cover both aspects and nicely show how software development goes along with theoretical research in the field of computational geometry today.

Let us briefly introduce the six papers of this issue. In *Almost-Delaunay simplices: Robust neighborhood relations for imprecise 3D points using CGAL*, Deepak Bandyopadhyay and Jack Snoeyink present an academic user perspective of CGAL. They describe their experience in turning a MATLAB prototype into a CGAL-based implementation for their intended application in computational biology. During the process, many different methods had to be tried. The authors describe how the flexibility of CGAL allowed to do this quickly and efficiently, due to its generic programming approach.

The second paper, *An adaptable and extensible geometry kernel* by Susan Hert, Michael Hoffmann, Lutz Kettner, Sylvain Pion and Michael Seel, describes the most important generic component of CGAL, the kernel. In CGAL, the kernel is a concept for which various models exist, and the paper describes generic programming techniques that make it easy to adapt existing kernel models, and to derive new ones. Through the kernel, the high-level logic of many algorithms in CGAL is separated from the low-level geometric primitives. It is therefore very easy to replace the primitives (according to varying speed and robustness requirements), by simply replacing the kernel model.

The idea of parameterizing algorithms and data structures with their primitive operations is predominant in CGAL. The third paper *Advanced programming techniques applied to CGAL's arrangement package* by Ron Wein, Efi Fogel, Baruch Zukerman and Dan Halperin shows how this is done in CGAL for planar arrangements. An arrangement is the partition of the plane into vertices, edges, and faces induced by a set of curves. Arrangements are ubiquitous in many applications, such as computer-aided design and manufacturing, cartography, and robot motion planning. Arrangements have been in CGAL from a very early stage, and they have gone through many redesigns and improvements. While the early implementations were essentially proofs of concept, the present design is to a large extent motivated by real applications. This development nicely mirrors the progress of computational geometry software on the whole over the last ten years.

The fourth paper, *Boolean operations on 3D selective Nef complexes: Data structures, algorithms, optimized implementations and experiments* by Peter Hachenberger, Lutz Kettner and Kurt Mehlhorn demonstrates the power of the *exact geometric computation* paradigm in solid modeling. This paradigm completely avoids numerical problems by error-free arithmetic operations on the lowest level, and error-free geometric primitives on a higher level. This requires the use of advanced (multiprecision, filtered, and algebraic) number types and is potentially much slower than computing with floating-point numbers. But in the paper, the authors show that exact geometric computation *can* be competitive, even in comparison with commercial CAD systems.

Paper number five, *A generic software design for Delaunay refinement meshing* by Laurent Rineau and Mariette Yvinec describes a CGAL package that realizes a meta-algorithm for meshing. The motivation is that many meshing algorithms start by computing a Delaunay triangulation of a suitable set of vertices from the domain to be meshed,

and then they refine this triangulation in order to meet certain criteria. All these algorithms essentially differ only in their refinement rules and can therefore be parameterized by these rules. Technically, such a parameterization is not obvious and must consist of several layers to be applicable in practice. This is the contribution of the paper, and the resulting package fits the generic programming philosophy of CGAL very well.

The sixth and last paper, *A package for exact kinetic data structures and sweepline algorithms* by Daniel Russel, Menelaos Karavelas and Leonidas Guibas shows how quickly even more recent concepts in computational geometry can make their way into quality software these days. Kinetic data structures have come up only in the late nineties, and the development of the basic theory is by no means complete. Still, the authors offer a generic package for implementing kinetic data structures under the exact geometric computation paradigm, and as a proof of concept, they show how this works for Delaunay triangulations of moving points.

In a sense, this last paper points into the future; new trends and techniques will emerge in computational geometry, both in theory and in software. For CGAL this shall result in fundamental geometric computing packages, as well as in packages in applied domains such as computer graphics, scientific visualization, computer aided design and modeling, geographic information systems, molecular biology, medical imaging, robotics and motion planning, mesh generation, multimedia retrieval, and numerical methods. The editors hope that CGAL will develop accordingly and remain a leading platform for geometric computing in practice, and a versatile tool in computational geometry teaching and research.

The reviewers for this special issue were selected from both within and outside the CGAL community. We thank them sincerely, this special issue would not exist without them.

December 2006

Bernd Gärtner
ETH Zurich, Switzerland

Remco Veltkamp
Utrecht University, Netherlands
E-mail address: Remco.Veltkamp@cs.uu.nl

Available online 12 February 2007