LiveCG: an Interactive Visualization Environment for Computational Geometry

Sebastian Kürten
Institut für Informatik, FU Berlin
sebastian.kuerten@fu-berlin.de

Wolfgang Mulzer
Institut für Informatik, FU Berlin
mulzer@inf.fu-berlin.de

ABSTRACT
We describe LiveCG, an interactive visualization environment for algorithms and data structures from Computational Geometry. LiveCG offers many primitives that make it easy to create interactive algorithm animations and illustrations for education and research. It can be seen as an attempt to develop a modern version of technologically obsolete systems such as XYZ Geobench or the Workbench for Computational Geometry.

1. INTRODUCTION
Algorithm animation has been very popular in the past, particularly in computational geometry. Nowadays, however, visualizations have become increasingly rare [10], and many old animations have disappeared along with the software used to create them. Historically, there were many visualization systems for computational geometry, such as GeoLab [4], XYZ GeoBench [9], Workbench for Computational Geometry [5], and GASP [11]. None of these are readily available today.

We regret this development, and we see the need for a new visualization platform using modern technology. To ease the task for developers, and to make the results widely accessible for users, we have developed the LiveCG framework.

2. DESIGN GOALS
The main design goals of LiveCG are usability, flexibility, modularity, and extendability. For developers, the framework provides a programming library for the creation of visualizations. It includes abstract data types for geometric objects, frequently used data structures, and implementations of basic geometric predicates and operations.

For users of the visualizations, there is an editor to create inputs for the different algorithms in a consistent fashion. This also relieves programmers from the burden of developing such a component for every new visualization.

The resulting visualizations are useful in many different settings. Once a valuable visualization has been created, it

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

SoCG’14, June 8–11, 2014, Kyoto, Japan.
Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM 978-1-4503-2594-3/14/06 ...$15.00.

Figure 1: The Geometry Editor.

3. THE FRAMEWORK
The framework is implemented in Java to support all major desktop platforms out of the box and to make it easy to port existing visualizations and applets. The programming library has abstract data types for points, polygonal chains, and polygons. The Geometry Editor lets users create these objects; see Fig. 1. It also allows for experimentation with the algorithms. Geometric objects can be created, modified, saved, and loaded, and animations can be launched from here. The framework already includes a database of example geometries for the implemented algorithms. It can be accessed through the editor’s main menu.

The programming library also includes implementations for rectangles, line segments, the DCEL, and triangulations, among others. There are also bidirectional bindings to the JTS Topology Suite and the geometry packages from AWT. This provides access to various geometric operations, such as spatial predicates, Boolean operations with polygonal objects, buffering, area and length measurements, geometry simplification, and spatial indexing.

We designed an abstraction layer for the implementation of graphical visualizations to support their applicability for...
different purposes. This is essentially a drawing interface against which the visualizations are programmed. We provide several back ends for this layer. Currently, the framework supports the creation of different types of raster images (PNG, JPEG, etc.), SVG images, TikZ figures and Ipe files.

In addition to the GUI, the framework comes with a command line interface (CLI) that can be used to create snapshots of algorithm animations in the aforementioned file formats. Even though this export functionality is also available through the graphical menu, the CLI provides some extra comfort. In particular, it can be used for scripting, e.g., when generating a website or when creating figures for a publication or presentation.

For educational purposes, visualizations can implement an explanation interface. This allows the framework to display explanatory information about the actions of an algorithm along with its graphical depiction.

4. EXISTING VISUALIZATIONS

The framework already provides a number of visualizations for algorithms and data structures. The current list includes the following:

**DCEL.** This visualization can be used in a standalone fashion to examine the DCEL of an arrangement. It can also be a building block for more advanced visualizations.

**Polygon Triangulation.** A static visualization of the algorithm by Garey et al. [7] that partitions a polygon into monotone pieces and then triangulates each piece to obtain a triangulation of the polygon.

**Shortest Paths in Polygons.** An interactive animation of the algorithm by Lee and Preparata [8] that computes the shortest path between two nodes inside a polygon. The diagonal-wise progress of the algorithm can be examined as the shortest path between two nodes inside a polygon. The DCEL visualization has been integrated to elucidate the details of the construction of the DCEL representation of the Voronoi cells.

**Chan’s Algorithm.** The visualization provides an animation of the gift-wrapping step of Chan’s optimal convex hull algorithm [3]. It takes as input a set of convex hulls and computes the Voronoi diagram by walking around the convex hulls in parallel. The animation presents for each vertex of the resulting convex hull how the algorithm determines the tangent to one of the small hulls that maximizes the emerging hull’s interior angle.

**Free Space Diagram.** The free space diagram, relevant for the classic algorithm for computing the Fréchet distance by Alt and Godau [1], can be examined for two polygonal chains. The three dimensional terrain is displayed as a two-dimensional heat-map by mapping each height to a distinct color.

5. FUTURE WORK

We hope that more visualizations will be implemented in the future to cover more basic algorithms and data structures of the field. The goal would be to have a catalogue that includes most of the topics taught in an introductory course on computational geometry, so that the framework can be used as a thorough instructional aid.

Care has already been taken to make it possible to create a browser-based back end by utilizing the Google Web Toolkit (GWT). A future goal is to implement such a back end to enable the creation of interactive web-based learning resources for computational geometry.

6. REFERENCES


