Implementation of an Efficient Constraint Solver for the Layout of Graphs in Delaunay*

Isabel F. Cruz and Donald I. Lambe

Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609, USA
{ifc,dejobaan}@cs.wpi.edu

Delaunay [2] is a visualization system to display and query object-oriented databases. It supports DOODLE [1], a visual and declarative meta-language, with which users can specify the display of quantitative information (such as bar charts and pie charts) and of qualitative information (such as graphs). The visual vocabulary available to the user comprises only simple visual primitives such as box, circle, and line. It also supports constraints as a means of specifying distances between objects (length constraints) and occlusion between objects (overlap constraints).

Work in graph drawing has traditionally focused on algorithmic approaches, which are designed to solve efficiently a limited class of graphs using a particular drawing. Less common approaches that allow the user to specify *what* the graph will look like, instead of *how* that drawing is going to be achieved are called *declarative*. Typically these approaches use constraints and are computationally inefficient [4].

We concentrate on the capability of DOODLE for specifying the drawing of graphs declaratively using the Delaunay system that we have been implementing. Besides addressing efficiency concerns, the solver should also deal with constraint cycles satisfactorily. For example, the SkyBlue [5] constraint solver actually fails to solve relatively simple constraints if they contain cycles, which are difficult to avoid in graph drawing problems.

We have therefore designed and implemented our own constraint solver, based on the algorithm presented in [3]. In [3], we showed that using DOODLE, we can specify a variety of drawings for a class of graphs that includes trees, series-parallel graphs, and acyclic digraphs. Furthermore, we have shown that such drawings can be achieved in optimal time.

The role of the constraint solver is to calculate the absolute locations and dimensions of each visual object. The coordinates of these objects are represented by *landmarks*, the horizontal or vertical distance between any two landmarks being represented by a *length constraint*. The constraint solving strategy is as

* Research supported in part by the National Science Foundation under CAREER Award IRI–9896052 and CISE Research Instrumentation Grant 9729878.

S.H. Whitesides (Ed.): GD'98, LNCS 1547, pp. 436–437, 1998.

follows. We build a constraint graph where each vertex represents a variable associated with a landmark, and each edge represents a dependency between two variables established by a constraint. The edges are directed for a *max* or *min* constraint, and are undirected otherwise for all other length constraints.

First, we need the absolute horizontal and vertical coordinates of an origin landmark. From these coordinates, the solver attempts to calculate the positions of other landmarks that share a length constraint with the origin. In subsequent iterations, the values of the coordinates of the newly positioned landmarks are propagated, until all variables are instantiated. We have successfully solved constraints where cycles are present and efficiency is guaranteed by avoiding duplication of computations. Like the rest of the Delaunay prototype, the constraint solver is implemented using Java.

We will be presenting details of the implementation, DOODLE programs that implement a variety of drawings (e.g., planar upward, containment, and H-V drawings of trees, Δ-drawings of series-parallel digraphs), and drawings as obtained by Delaunay.

References

[1] I. F. Cruz. DOODLE: A Visual Language for Object-Oriented Databases. In *ACM-SIGMOD Intl. Conf. on Management of Data*, pages 71–80, 1992.

[2] I. F. Cruz et al. Delaunay: a Database Visualization System. In *ACM-SIGMOD Intl. Conf. on Management of Data*, pages 510–513, 1997.

[3] I. F. Cruz and A. Garg. Drawing Graphs by Example Efficiently: Trees and Planar Acyclic Digraphs. In *Graph Drawing '94*, number 894 in Lecture Notes in Computer Science, pages 404–415. Springer Verlag, 1995.

[4] E. Dengler, M. Friedell, and J. Marks. Constraint-Driven Diagram Layout. In *IEEE Symposium on Visual Languages*, 1993.

[5] M. Sannella. The SkyBlue Constraint Solver. Technical Report 92-07-02, Computer Science Department, University of Washington, February 1992.