3DCube: a Tool for Three Dimensional Graph Drawing*

Maurizio Patrignani[1] and Francesco Vargiu[2]

[1] Dipartimento di Informatica e Automazione,
Università di Roma Tre, Rome, Italy
patrigna@inf.uniroma3.it
[2] Autorità per l'Informatica nella Pubblica Amministrazione, Italy
vargiu@aipa.it

Abstract. In this paper we describe a tool that is a general frame for the three-dimensional representation of graphs, especially devoted to the algorithms evaluation, refinement and development. 3DCube (3D Diagram Drawer) offers innovative features in the user interaction and contains a set of three-dimensional algorithms both taken from the literature and proposed by the authors.

1 Introduction

Three dimensional graph drawing is an emerging field in the graph drawing area. Several tools are already available for the representation of graphs in the plane, and the most sophisticated of them also allow some three-dimensional representation, either as an additional presentation feature of basically $2D$ results (see GMB [13] and PLUM [17]), or as the result of an actual $3D$-drawing algorithm (see 3DSA [4], COMAIDE [6], the ffGraph library [9], GEM-3D [2], GIOTTO3D [11], GOLD [10], GOVE [18]), and PARSA [14]).

The tool we describe in this paper offers to the user a general frame for the representation of graphs entirely devoted to $3D$-drawing algorithms, and especially to their evaluation, development and refinement. It complies to the requirements of an algorithm independent tool, by supporting a set of $3D$-algorithms taken from the literature and proposed by the authors. Furthermore 3DCube proposes an innovative interaction paradigm to the user, both for graph browsing and for inspection and understanding of the layout process.

2 Requirements and Functionalities

The aim of 3DCube is not restricted simply to a representation system. The tool provides also a support i) to the visual analysis of the behavior of the layout algorithms during their execution; ii) to the refinement of algorithms, especially those based on heuristics or including tunable steps; and iii) to the direct comparison among different algorithms.

The following list summarizes the main features of 3DCube:

* Research supported in part by the ESPRIT LTR Project no. 20244 - ALCOM-IT

$3D$-perception. We used any known feature to allow the user to recognize distances and dimensions of the $3D$ objects mapped on the screen, so enhancing the comprehension of object distribution in space.
The user of 3DCube is able to browse through the representation, by using the three primitives rotation, shifting, and zooming, which allow to simulate any real movement. Depth cuing, perspective viewing and surface characteristics are used to obtain a realistic $3D$ effect.
The tool implements also a *Snapshot* utility that allows to store a collection of meaningful views of a diagram. By selecting a view from a film-like sequence, the user asks the tool to recreate the same conditions in which the snapshot was captured, improving the user-friendliness of the system.

Parametric Representation. Different shapes and properties for nodes and edges can be specified according to the user requirements.
We adopt a separation between the abstract description of the graphs and the geometrical properties of nodes and edges; we use a synthetic and adaptable model to describe them, compatible with the file formats of Diagram Server [5]. A list of graphic primitives (such as spheres, cubes, polygons) and associated attributes (colors, dimensions, etc.) defines each type of node. Analogously, for each type of edge it is possible to define the shape of bends and of segments.

Algorithms Visual Inspection and Understanding. By understanding the operations involved in a layout algorithm and how such operations reflect in the final results, the user is supported to possibly improve the algorithm itself or to conceive new algorithms.
3DCube offers an *algorithm animation* facility that allows to inspect the intermediate results of a diagram construction by means of its graphical representation. This facility permits to animate the passage from consecutive representations (both forward and backwards), so that the user may easely follow the evolution of the drawing through a continuous movement of the involved objects.

Extensibility. The capability of adding new algorithms and new features without a main implementation effort is achieved by using a modular approach in the architecture design.

3 The Architecture of 3DCube

The architecture of 3DCube was conceived according to an object oriented methodology. It consists of a set of independent modules that exchange informations and services to each other, as shown in Figure 1.

The *User Interface* manages the interaction with the user by means of the windows environment, menus, buttons and controls. A particular attention was placed to support the navigation (a control panel makes available different features) and to support the depth perception, using perspective, shades and colors.

The *Graphic Driver* filters each graphic system call from the User Interface to the computing platform thus improving the portability of the tool.

3DCube is not limited to the showing of the results of a specific layout algorithm. The *Algorithm Manager* is the repository of algorithms; in the current version it mainly contains $3D$-orthogonal drawing algorithms. We cite the algo-

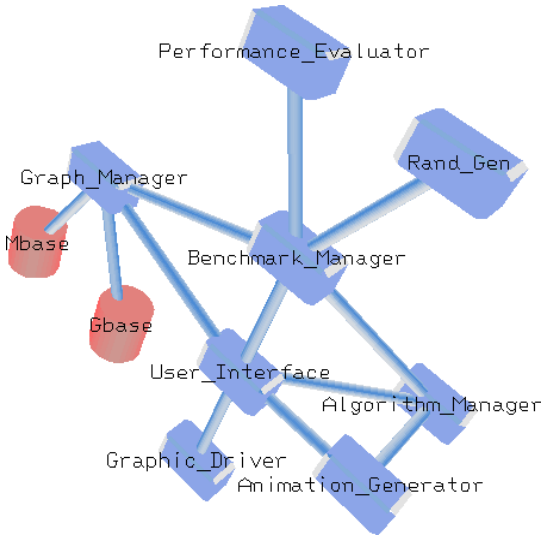

Fig. 1. The architecture of 3DCube.

rithm proposed by Therese Biedl [1], the Komolgorov and Bardzin algorithm [7], the Compact and the Three Bends [8], the algorithm proposed by Papakostas and Tollis [15], and Reduce Forks, proposed by the authors [16]; moreover it contains the $3D$-straight-line drawing algorithm, called Momentum Curve, of Cohen, Eades, Lin and Ruskey [3].

The *Animation Generator* module, by using the services of the Algorithm Manager, is able to simulate the continuos movement of nodes and edges during the algorithm execution.

The *Graph Manager* implements the storing and loading functions for the graphs, their representations, the snapshots, and the associated models in the *GBase* and *MBase*.

The *Benchmark Manager* allows the automatic execution of a list of algorithms on a set of graphs, and is able to generate reports about the behavior of the algorithms with respect to a given set of aesthetic criteria.

For this purpose the Benchmark Manager interacts with the *RandGen* module which is able to generate a base of random graphs according to some user-specified features. It also communicates with the *Performances Evaluator* to obtain the computation of the results with respect to the selected criteria and to generate reports and graphics.

4 An example of graph browsing

In this section we describe, with some pictures, the main features of the user interface of 3DCube, by simulating a typical work session. The user may load and visualize one or more different graphs at the same time. For every loaded graph 3DCube shows on a window the list of the drawing methods applyable to the

given graph (see Figure 2). The selection of a method triggers its execution on the graph and the obtained representation is displayed on a new window, called the *diagram window*. In the above mentioned picture it is shown a diagram window with the result of Reduce Forks applied to a graph with 24 nodes.

Figure 3 shows a K_6 graph drawn by using the algorithms of Biedl, Compact, Komolgorov and Bardzin, Papakostas and Tollis, Reduce Forks, and Three Bends, respectively. The control panel allows the user to navigate through the representation to inspect the objects displacement. During the navigation, the user can save meaningful views of the representation by clicking on the snapshot button. The collection of all previously saved snapshots for the given representation is shown in the snapshot window (see Figure 2).

5 An example of drawing algorithm refinement

In this section we describe how 3DCube was effectively used in the development of a new $3D$-orthogonal drawing algorithm.

The starting idea was to obtain a $3D$-orthogonal drawing through the following iterative procedure: firstly the same coordinates are assigned to all the nodes of the graph; then at every step a plane (orthogonal to one of the axes and hosting at least two nodes with the same coordinates) is split in two, and its nodes are redistributed between the two resulting planes. The procedure terminates when all nodes have different coordinates. The procedure sketched above is not sufficient to produce an orthogonal drawing: when a split occurs, an edge may become slant. So slant edges are split in two, as they appear, and a dummy node (that eventually will become a bend) is inserted and positioned in such a way that its two adjacent edges are orthogonal. Furthermore, we defined four sufficient conditions, that, once they are satisfied one after the other, guarantee the termination of the algorithm, namely:

- the separation between original nodes,
- for each original node, the coherence of the directions of its incident edges,
- the separation between original and dummy nodes, and
- the separation between dummy nodes.

In the first version of the algorithm, called Four Targets, we defined and implemented four different split criteria, based on heuristics, each devoted to satisfy one of the four corresponding conditions of the previous list.

By extensively using the tool in order to verify the effectiveness of Four Targets we noticed that, while the algorithm satisfied the convergence conditions, it made very costly choices with respect to the final number of bends and volume occupation. So we reviewed the last three split criteria by introducing a ranking among alternative split operations, aimed to minimize the number of bends introduced. The new algorithm significantly outperformed the previous one.

Finally we noted that a specific configuration, called *fork*, consisting of a pair of adjacent edges both with an end in each of the plane generated by a split, was responsible for the insertion of an extra bend either at the end of the split operation, or in a successive split. Thus we conceived a new heuristic for the first split criterium, aimed to reduce the number of forks.

Experiments showed that the performance of this method, called Reduce Forks, are interesting with respect to the number of bends and space occupation. By using 2000 random test graphs approximately, with a number of nodes between 10 and 100, we obtained an average of less than 2.5 bends per edge and a volume occupation of $0.6n^3$.

While the drawings produced by the Reduce Forks algorithm are, in our opinion, aesthetically pleasant, we are aware that the algorithm may be further improved both by tuning the existent steps, and by possibly adding further steps (such as a bend stretching post processing step).

6 Conclusions and Future Improvements

The current version of the tool is developed in C++ language, on a UNIX RISC 6000 workstation, using the Motif libraries for the management of the windows environment and the graPHIGS libraries for the three-dimensional visualisation. We plan to rewrite the tool by using the VRML language to make available the implementation of the graph drawing algorithms on the Internet. We aim to increase the number of implemented graph drawing algorithms. Finally, in order to increase the interoperability with other tools, we aim to add support for new formats such as GML [12].

Acknowledgements

We are grateful to Giuseppe Di Battista for initiating our interest in the $3D$ graph drawing area, for his comments, suggestions and encouragement. We are also grateful to Walter Didimo, Antonio Leonforte, Sandra Follaro and Loredana Bottaro for their support.

References

1. T.C. Biedl. Heuristics for 3D-Orthogonal Graph Drawing. Presented at the 4th Twente Workshop, Enschede, June 1995. `http://rutcor.rutgers.edu:80/ therese/ps/3D.twente.ps`
2. I. Bruß and A. Frick. Fast Interactive 3-D Graph Visualization. *Proc. GD '95*, LNCS 1027, pp. 99-110, Springer-Verlag, 1995. `ftp://i44ftp.info.uni-karlsruhe.de/pub/papers/frick/gd95p.ps.gz`
3. R. F. Cohen, P. Eades, T. Lin and F. Ruskey. Three-dimensional graph drawing. *Proc. GD '94*, LNCS 894, pp. 1-11, 1994.
4. I.F. Cruz and J.P. Twarog. 3D Graph Drawing with Simulated Annealing. *Proc. GD '95*, LNCS 1027, pp. 162-165, Springer-Verlag, 1995.
5. G. Di Battista, G. Liotta, and F. Vargiu. Diagram Server. *JVLC* (special issue on Graph Visualization, I. F. Cruz and P. Eades, editors), 6(3), 1995.
6. D. Dodson. COMAIDE: Information Visualization using Cooperative 3D Diagram Layout. *Proc. GD '95*, LNCS 1027, pp. 190-201, Springer-Verlag,1995.
7. P. Eades, C. Stirk and S. Whitesides. The Techniques of Komolgorov and Bardzin for Three Dimensional Orthogonal Graph Drawing. *TR 95-07, Dept. of Computer Science, University of Newcastle NSW, Australia*, October 1995. `ftp://ftp.cs.newcastle.edu.au/pub/techreports/tr95-07.ps.Z`

8. P. Eades, A. Symvonis and S. Whitesides. Two Algorithms for Three Dimensional Orthogonal Graph Drawing. *Proc. GD '96*, LNCS 1190, pp. 139-154, Springer-Verlag,1996.
9. C. Friedrich. The ffGraph Library. *Lehrstuhl für Informatik*, Universität Passau, December 1995. `http://kaolin.unice.fr/Documentation/Doc.html`
10. A. Frick, C. Keskin and V. Vogelmann. Integration of Declarative Approaches (System Demonstration). *Proc. GD '96*, LNCS 1190, pp. 184-192, Springer-Verlag,1996.
11. A. Garg and R. Tamassia. GIOTTO3D: A System for Visualizing Hierarchical Structures in 3D. *Proc. GD '96*, LNCS 1190, pp. 193-200, Springer-Verlag, 1996.
12. M. Himsolt. The Graphlet System (System Demonstration). *Proc. GD '96*, LNCS 1190, pp. 233-240, Springer-Verlag, 1996.
13. D. Jablonowsky and V.A. Guarna. GMB: A Tool for Manipulating and Animating Graph Data Structures. *Softw. - Pract. Exp.*, 19(3), pp. 283-301, 1989.
14. B. Monien, F. Ramme and H. Salmen. A Parallel Simulated Annealing Algorithm for Generating 3D Layouts of Undirected Graphs. *Proc. GD '95*, LNCS 1027, pp. 396-408, Springer-Verlag, 1995.
15. A. Papakostas. Information Visualization: Orthogonal Drawings of Graphs. *PhD thesis*, Department of Computer Science, University of Texas at Dallas, Dec. 1996.
16. M. Patrignani. Visualizzazione di Diagrammi in Tre Dimensioni. *Ms. Sc. Degree thesis*, Dipartimento di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza", Oct. 1996.
17. S.P. Reiss. E-D Visualization of Program Information. *Proc. GD '94*, LNCS 894, pp. 12-24, Springer-Verlag, 1995.
18. R. Webber and A. Scott. GOVE Grammar-Oriented Visualisation Environment. *Proc. GD '95*, LNCS 1027, pp. 516-519, Springer-Verlag, 1995.

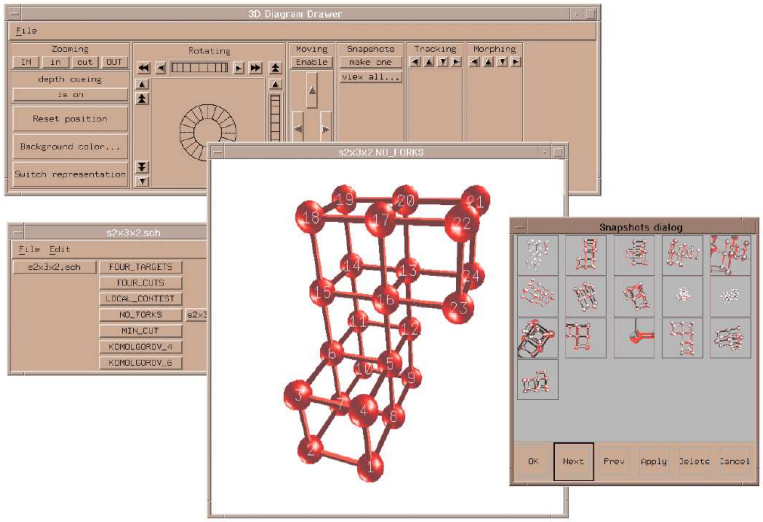

Fig. 2. A session of 3DCube displaying a graph of 24 nodes drawn by the Reduce Forks algorithm and the snapshot window.

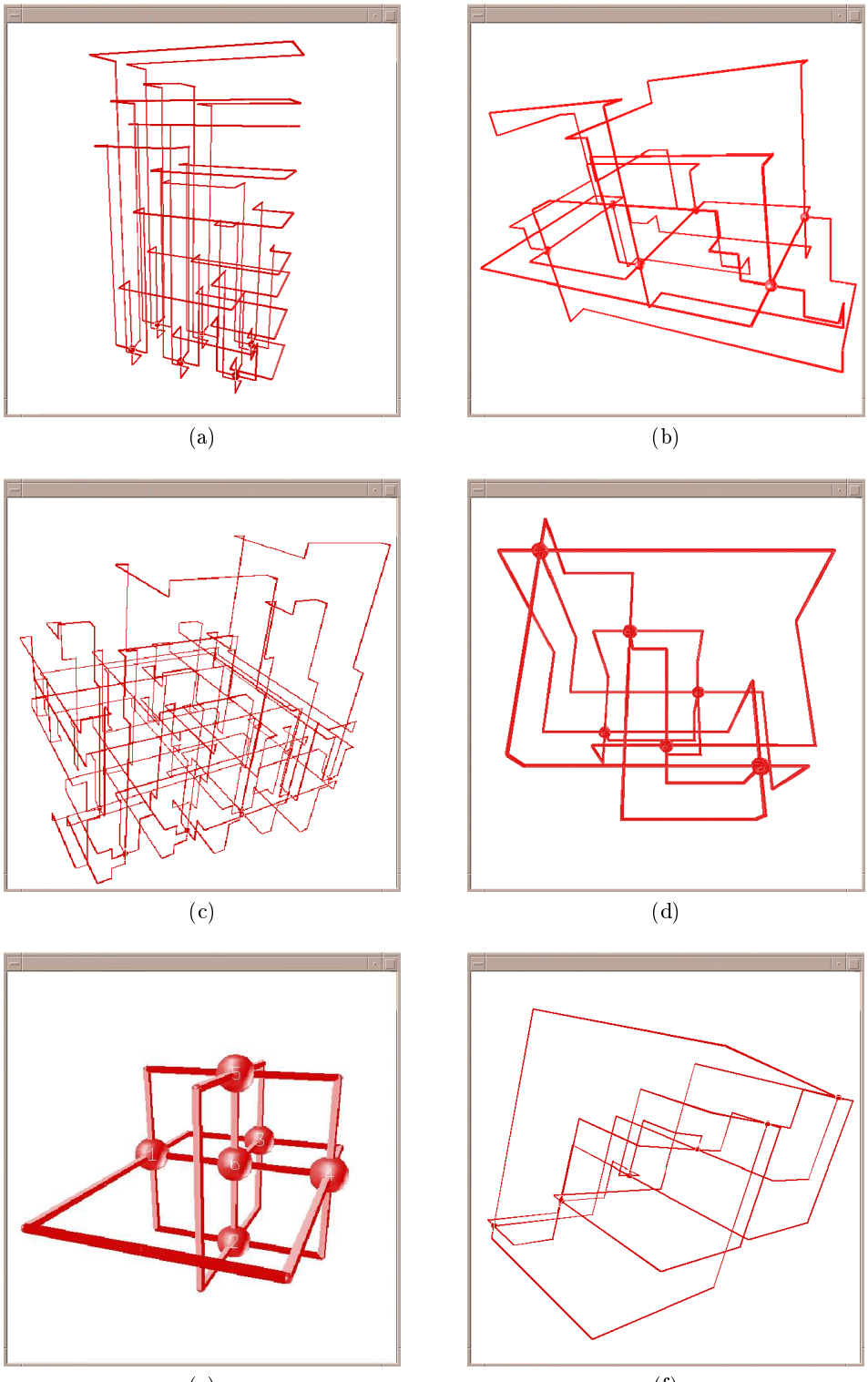

Fig. 3. The Biedl (a), Compact (b), Komolgorov (c), Papakostas-Tollis (d), Reduce Forks (e), and the Three Bends (f) algorithms applied to a K_6 graph.