Journal of Graph Algorithms and Applications

http://www.cs.brown.edu/publications/jgaa/

vol. 4, no. 3, pp. 183–191 (2000)

Dynamic WWW Structures in 3D

Ulrik Brandes *Vanessa Kääb* *Andres Löh* *Dorothea Wagner* *Thomas Willhalm*

Department of Computer & Information Science
University of Konstanz
78457 Konstanz, Germany
http://www.inf.uni-konstanz.de/
~{brandes,kaeaeb,loeh,wagner,willhalm}
{Ulrik.Brandes, Vanessa.Kaeaeb, Andres.Loeh, Dorothea.Wagner, Thomas.Willhalm}@uni-konstanz.de

Abstract

We describe a method for three-dimensional straight-line representation of dynamic directed graphs (such as parts of the World Wide Web). It has been developed on the occasion of the 1998 Graph Drawing Contest and constitutes a customized blend of techniques known from the Graph Drawing literature. Since we feel that they may be of interest to others facing similar graph drawing problems, some technical details are mixed in.

The animation of the contest graph and accompanying material are available from the *Journal of Graph Algorithms and Applications*'s Web site.

Communicated by G. Liotta and S. H. Whitesides: submitted November 1998, revised October 1999.

1 Introduction

Since 1993, a contest is organized along with the annual *Symposium on Graph Drawing* [7, 8, 10, 11, 9]. It usually features graphs from different applications posing distinct problems with respect to graphical presentation. In the present case [9], a dynamic graph of links between World Wide Web (WWW) pages was given as a list of link additions and deletions, shown in Figure 1. The contestants should

> "...depict the content and structure of the graph as it evolves."

Our primary interest was to study the implied dynamic graph layout problem. Hence, we largely ignored content and focussed on evolving structure.

```
add www.att.com/catalog/consumer -> www.att.com
add www.att.com/att -> www.att.com/catalog/consumer
add www.att.com -> www.att.com/catalog/consumer
add www.att.com/catalog/consumer -> www.att.com/cmd/custcare
add www.att.com/catalog/consumer -> www.att.com/write
add www.att.com -> www.att.com/worldnet
add www.att.com/catalog/consumer -> www.att.com/terms.html
add www.att.com/att -> www.att.com/learningnetwork
add www.att.com/catalog/consumer -> www.att.com/cgi-bin/ppps.cgi
add www.att.com -> www.att.com/catalog/small_business
add www.att.com/whatsnew -> www.att.com/catalog/consumer
add www.att.com/catalog/small_business -> www.att.com/cgi-bin/bmd_cart.cgi
add www.att.com/catalog/consumer -> www.att.com/cmd/jump
add www.att.com/att -> www.att.com/catalog
add www.att.com/textindex.html -> www.att.com/catalog/small_business
add www.att.com/catalog/small_business -> www.att.com/bmd/tollfree
add www.att.com/whatsnew -> www.att.com/worldnet/wmis
add www.att.com/catalog/consumer -> search.att.com
add www.att.com/catalog/small_business -> www.att.com/bmd/jump
add www.att.com/features -> www.att.com/rock
add www.att.com/catalog/small_business -> www.att.com
add www.att.com/home -> www.att.com/catalog/consumer
delete www.att.com -> www.att.com/catalog/consumer
add www.att.com/catalog/small_business -> www.att.com/services
delete www.att.com/att -> www.att.com/learningnetwork
delete www.att.com/catalog/consumer -> www.att.com/cmd/custcare
add www.att.com/catalog/small_business -> www.att.com/write
add www.att.com/net -> www.att.com/worldnet/wis/sky/signup.html
add www.att.com/catalog/small_business -> www.att.com/bmd/products
add www.att.com/catalog/small_business -> search.att.com
add www.att.com/net -> www.att.com/worldnet/wmis
add www.att.com/catalog/small_business -> www.att.com/terms.html
delete www.att.com/catalog/small_business -> www.att.com/bmd/tollfree
add www.att.com/att -> www.att.com/catalog/small_business
add www.att.com/news -> www.att.com/catalog/consumer
add www.att.com/whatsnew -> www.att.com/catalog/small_business
add www.att.com/net -> www.att.com/worldnet/intranet
delete www.att.com/whatsnew -> www.att.com/catalog/consumer
add www.att.com/catalog/small_business -> www.att.com/bmd/custcare
add www.att.com/catalog/consumer -> www.att.com/cgi-bin/cart.cgi
delete www.att.com/home -> www.att.com/catalog/consumer
add www.att.com/catalog/small_business -> www.att.com/cmd
add www.att.com/catalog/consumer -> www.att.com/services
add www.att.com/worldnet -> www.att.com/worldnet/intranet
add www.att.com/services -> www.att.com/catalog
delete www.att.com/catalog/consumer -> www.att.com
add www.att.com/services -> www.att.com/catalog/consumer
delete www.att.com/catalog/consumer -> www.att.com/services
add www.att.com/services -> www.att.com/catalog/small_business
add www.att.com/whatsnew -> www.att.com/news
delete www.att.com/whatsnew -> www.att.com/worldnet/wmis
add www.att.com/catalog/consumer -> www.att.com/cmd/products
delete www.att.com/net -> www.att.com/worldnet/wis/sky/signup.html
delete www.att.com/catalog/small_business -> www.att.com/services
add www.att.com/speeches -> www.att.com/speeches/index96.html
add www.att.com/worldnet -> www.att.com/worldnet/wmis
delete www.att.com/att -> www.att.com/catalog
add www.att.com/textindex.html -> www.att.com/catalog/consumer
delete www.att.com/services -> www.att.com/catalog
add www.att.com/news -> www.att.com/catalog/small_business
add www.att.com/international -> www.att.co.uk
delete www.att.com/catalog/small_business -> search.att.com
add www.att.com/business -> www.att.com/catalog/small_business
delete www.att.com/textindex.html -> www.att.com/catalog/consumer
add www.att.com/write -> www.catalog.att.com/cmd/jump
```

Figure 1: Data given for Graph A of the 1998 Graph Drawing Contest

By definition the graph starts out empty, and directed edges, sometimes introducing new vertices, are added and deleted. To get a first impression of

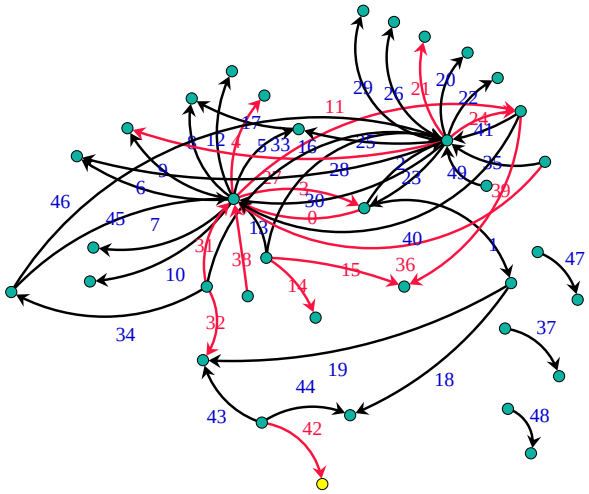


Figure 2: 2D representation with insertion (black edges) or deletion (gray edges) times. Edge curvature indicates the order of creation of incident vertices

the graph's structure, in particular of its size and density, it was input in a graph editor (LEDA's [12] GraphWin), with edges colored differently, if they are inserted and later deleted. See Figure 2.

We found that the graph was sufficiently small and sparse to try animating a straight-line representation in space. For such an animation we essentially had to cope with three aspects, each of which is addressed in a subsequent paragraph. First of all, a dynamic layout model had to be devised, specifying vertex trajectories. Secondly, unlike in two-dimensional representations, a point from which to view the graph had to be selected carefully, because otherwise substantial parts of the structure might be occluded. And finally, a rendering had to be specified. In Section 5 we conclude with lessons learned from this project.

2 Dynamic Graph Layout

Since no dynamic three-dimensional straight-line layout algorithm was known to us, we did our own experiments using random field layout models, a flexible modeling scheme introduced in [1]. A prototypical implementation of a corresponding layout module was available to us because of its use in other projects.[1]

Our dynamic layout model consists of a sequence of static layouts and is hence described in two steps. First, structures are mapped into space by static layout models, which are then extended into a sequence of dependent layouts, in sync with graph modifications.

[1] One of these projects is described in this issue [3], together with a brief overview of the modeling framework.

Static layout model. Suppose we are given a single directed graph $G = (V, A)$, and suppose further that vertices ought to be represented by points in space, while edges ought to be represent by straight lines. It is then sufficient to compute a vector $p = (p_v)_{v \in V}$ of vertex positions. Any such vector is called a *layout*. An objective function defined on p is used to formally capture the quality of a layout, defined by local criteria such as uniform vertex distribution, uniform edge lengths, and sufficient distance between vertices and those edges they are not incident to.

The corresponding terms of our objective function (see Table 1), which is very similar to [5], are called *potentials*, since the objective function itself can be interpreted as an *energy* measuring layout distortion with respect to the above criteria. The additional potentials account for the contest graph being directed and, most of the time, disconnected. To avoid that disconnected components drift arbitrarily far apart, one representative vertex of each component is connected to the layout space origin by an invisible edge.[2] Even though WWW links form a general directed graph, they tend to contain hierarchical, tree-like substructures (if purely navigational links are omitted, as is the case in the contest data).[3] To encode the direction of edges graphically, we hence favor them pointing downward, thus displaying edge directions in context rather than locally by using cone-shaped arrows at the end of an edge. Note that this is similar to layered [14] and upward [6] layouts, however in 3D. Our rotation potentials are an adaptation of rotative forces used in [13] to align edges with an imaginary magnetic field. Note that squaring a normalized angle renders angles larger than $\frac{\pi}{2}$ (upward pointing edges) more severe a distortion than small deviations (downward pointing edges not exactly parallel to the the z-axis).

Dynamic layout model. The contest data implies a sequence $G^{(0)}, \ldots, G^{(65)}$ of directed graphs, where $G^{(0)}$ is the empty graph. To produce a smooth animation, a sequence of static layouts was computed. Vertex positions in these layouts were then used as break points of splines, i.e. smooth curves interpolating these points and thus defining vertex trajectories for the animation.

For every $G^{(t)}$ with $0 < t \leq 65$ we computed three layouts, $p^{(t,0)}, p^{(t,1)}, p^{(t,2)}$, in slightly different ways described below. For graceful animation it is necessary that consecutive layouts depend on each other. For simplicity, we let each layout depend only on its predecessor, so that the sequence of layouts has the form

$$
\begin{array}{llllllll}
\downarrow G^{(1)} & & & & & & & \\
p^{(1,0)} & \rightarrow p^{(1,1)} & \rightarrow p^{(1,2)} & & & & & \\
 & & \downarrow G^{(2)} & & & & & \\
 & & p^{(2,0)} & \rightarrow p^{(2,1)} & \rightarrow p^{(2,2)} & & & \\
 & & & & \downarrow G^{(3)} & & & \\
 & & & & p^{(3,0)} & \rightarrow p^{(3,1)} & \rightarrow p^{(3,2)} & \\
 & & & & & & & \vdots
\end{array}
$$

[2] For the contest graph, these vertices could be selected manually, since in each component there is at least one vertex that is present the entire life of the component. In general, one could use the vertex closest to the origin whenever the previous representative is deleted.

[3] Presumably because of hierarchical content organization and storage in likewise filesystems. An alternative considered briefly was therefore to use horizontal layers defined by subdirectories.

potential	definition	effect
repulsion	for each pair of non-adjacent vertices $u, v \in V$ evaluate $\frac{c_\delta^4}{d(p_u,p_v)^2}$	spreads vertices evenly in space
distance	for each pair of adjacent vertices $u, v \in V$ evaluate $\frac{c_\delta^4}{d(p_u,p_v)^2} + d(p_u,p_v)^2$	causes edges to have length roughly equal to c_δ
repulsion	for each pair $v \in V$ and $(u,w) \in A$ evaluate $\frac{c_\delta}{d(p_v;p_u,p_w)^2}$	prevents edges going through vertices
attraction	for a representative $v \in V$ of each component evaluate $d(p_v,(0,0,0))^2$	prevents components from drifting apart by tying them to the origin
rotation	for each $(u,v) \in A$ evaluate $c_\varrho \cdot c_\delta^3 \cdot \left(\frac{\arccos \frac{z_u - z_v}{d(p_u,p_v)}}{\frac{\pi}{2}} \right)^2$	favors edges pointing downward

Table 1: Building blocks of a static layout objective function for a directed graph $G = V, A)$, where $p_v = (x_v, y_v, z_v)$ is a point in space, $d(p_u, p_v)$ denotes the Euclidean distance between positions p_u and p_v, and $d(p_v; p_u, p_w)$ denotes the smallest Euclidean distance between p_v and any point on the straight line connecting p_u and p_w. We used $c_\delta = 60$ and $c_\varrho = 0.25$

The first layout of each graph $G^{(t)}$, $p^{(t,0)}$, was computed by minimizing the static layout objective function, subject to $p_v^{(t,0)} = p_v^{(t-1,0)}$ for those vertices v that are present in both $G^{(t)}$ and $G^{(t-1)}$. That is, the at most two newly introduced vertices are placed optimally in the unchanged previous layout.

It is argued in [2] that dependencies between layouts should be modeled by adding a stability term to the objective function. Good choices for stability terms are difference metrics [4] capturing the notion of change between two layouts. A natural candidate for straight-line embeddings is the sum of Euclidean distances between positions of corresponding vertices. To compute $p^{(t,i)}$, $i = 1, 2$, we hence augmented the static layout model with stability potentials

$$\sigma_i \cdot d\left(p_v^{(t,i)}, p_v^{(t,i-1)}\right)^2$$

for all vertices v, attracting vertices to their previous position. Constant σ_i controls the strength of the attraction and thus relative importance of stability. We used $\sigma_1 = 1.5$ and $\sigma_2 = 0.75$. For obvious reasons, this notion of stability is called *anchoring*, and it has an interesting probabilistic interpretation [2].

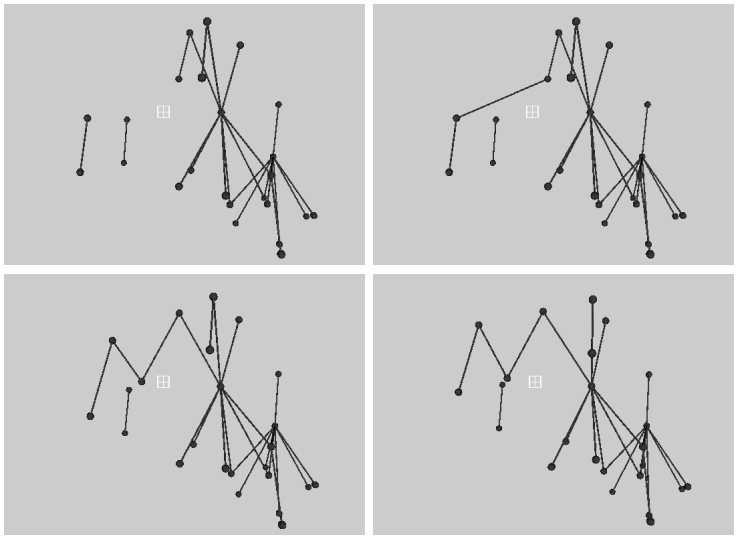


Figure 3: Insertion of an edge (directed left to right)

The resulting minimization problems were solved approximately by simulated annealing, an iterative method for combinatorial optimization, again because we already had the implementation available. Though simulated annealing is notoriously slow, its running time was negligible in this context (see below). Quality of the 195 layouts generated was controlled using simple VRML[4] output (ball-and-stick representations, see Figure 3).

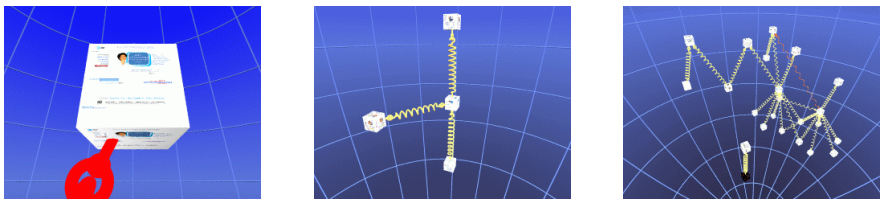
3 Viewpoints

There were no truly three-dimensional media available to display the animation. We therefore had to select two-dimensional projections of the graph, i.e. positions of an observer in space and his or her directions of view. For an overview of viewpoint selection we refer to [15]. Here, we used a simple heuristic to find a decent viewpoint under perspective projection once after each modification of the graph. In the course of the animation we let the observer follow a spline through these points while focusing on the origin.

Good viewpoints are those that are not bad. A viewpoint is said to be bad, if it yields a projection in which an item hides another one or false incidences are suggested. A projection is said to yield a vertex-vertex, a vertex-edge, or an edge-edge *occlusion*, if two vertices, a vertex and an edge, or a two edges coincide in the projection, but not in the three-dimensional layout. The random viewpoint of Figure 3 is bad.

For a given distance from the origin, we can imagine the observer to sit on a sphere centered at the origin. Recall that we let the observer focus on the origin at all times. A point on the sphere yields an occlusion, if it is collinear with either two vertices, a vertex and an edge, or two edges. Because the points on

[4]Virtual Reality Modeling Language, see `http://www.web3d.org/vrml/`. Since we had to use VRML 1.0, the effectiveness of the dynamic model was checked using a script that remotely loaded the output files into the VRML browser, one at a time.



(a) thumbnails (b) spatial relations (c) viewpoints

Figure 4: Three out of 3250 frames

the sphere that are collinear with any two points on a pair of non-coplanar edges cover large areas on the sphere, we ignore edge-edge occlusions altogether. In general we cannot expect to find any point on the sphere not yielding an edge-edge occlusion, anyway.

The *goodness* of a viewpoint is defined to be its minimum great-circle distance from any occlusion. This measure of goodness is called *rotational separation* in [15]. For the animation we approximated the best viewpoint under rotational separation within great-circle distance at most $\pi/2$ from the previous one to suppress large rotations, by placing an spherical grid on the observer's semi-sphere and exhaustively searching for the best grid point.

Finally, we fit the graph onto the screen by adjusting the radius of the observer sphere such that all vertices are just inside the finite projection area.

4 Rendering

As a small hint on the actual content of the graph, vertices were represented by boxes with corresponding WWW pages texture-mapped onto their sides (thumbnail representations).[5] Figure 4(a) shows the initial frame in which positions have been modified manually to yield a more attractive get-go. Springs are used to render edges, just because they are reminiscent of the physical analogy underlying the layout. It is a by-product that their coils also visualize the stress on an edge. It can be seen from Figure 4(b) that referred pages are hooked to their referring pages. In the case of bidirectional edges, just one spring is shown but both ends are hooked. Edges added or deleted in a modification are highlighted during that event. A spherical grid surrounding the graph helps to distinguish between vertex movements and movements of the viewpoint.

Spline break points and rendering information are specified in a POV-Ray[6] scene description file. Using the ray tracer, we generated 50 still frames per modification, which were subsequently converted into an animation with 25 frames per second (among others, in MPEG-1[7] format).

[5] One page could not be retrieved and is therefore represented all white. Towards the end of the animation, two boxes seem to not have thumbnails, but in fact the corresponding WWW pages have a black background.

[6] Persistence of Vison Ray Tracer, see `http://www.povray.org/`.

[7] Moving Picture Experts Group, see `http://drogo.cselt.stet.it/mpeg/`

5 Conclusion

The final animation can be retrieved from the accompanying WWW page.[8] Compared to the time it took to render all 3250 frames (one and a half nights on half a dozen workstations), layout and viewpoint selection times were negligible (35 minutes on a SUN Ultra-1 workstation, 80% for viewpoints). We believe that both layout and viewpoint selection can be improved to interactive speed. Since the layout objective function has no discrete terms, standard force-directed methods may be applied. Moreover, very few iterations are needed, since the initial layouts are by definition close to a locally optimal solution. Using newer features not present in VRML 1.0 and a sufficiently fast browser, we therefore do think that our approach can be customized for interactive applications. Given the convincing result of our off-line experiments, we definitely encourage research in this direction. The main open problem will be to deal with directed cycles, since, depending on the rest of the graph, they either cause several layers to be lifted onto one, or destroy the otherwise emergent layering altogether.

In working on this project, we found that the contest is a unique opportunity to experiment with graph drawing techniques and to spread knowledge of them. Three of the authors were students at that time, but certainly all of us have learned more about graph drawing and short term project management. The task nicely split into almost independent subtasks (layout, viewpoints, rendering) and most of the time pragmatic solutions would be preferred over quality or running time improvements. The restricted time span, gradual (visually comprehensible!) improvement of intermediate results, and the competitive nature of a contest combined to keep motivation high. It goes without saying that the party we were able to throw with the prize money (featuring a continuously running animation on a large canvas) was considered a rewarding finale.

Acknowledgments

We wish to thank the referees for greatly improving the presentation. Special thanks to the editors of this issue for giving us the opportunity to write this non-standard kind of paper.

References

[1] U. Brandes. *Layout of Graph Visualizations*. PhD thesis, University of Konstanz, 1999. See `http://www.ub.uni-konstanz/kops/volltexte/1999/255/`.

[2] U. Brandes and D. Wagner. A Bayesian paradigm for dynamic graph layout. G. Di Battista, editor. *Proc. 5th Intl. Symp. Graph Drawing (GD '97)*, *Lecture Notes in Computer Science* vol. 1353, pages 236–247. Springer, 1997.

[3] U. Brandes and D. Wagner. Using graph layout to visualize train interconnection data. In this issue.

[8] The animation submitted to the Graph Drawing Contest was rendered at 640×480 pixels. Due to its immense file size, a reproduction with 320×240 pixels is provided instead.

[4] S. Bridgeman and R. Tamassia. Difference metrics for interactive orthogonal graph drawing algorithms. In this issue.

[5] I. F. Cruz and J. P. Twarog. 3D graph drawing with simulated annealing. F. J. Brandenburg, editor. *Proc. 3rd Intl. Symp. Graph Drawing (GD '95)*, *Lecture Notes in Computer Science* vol. 1027, pages 162–165. Springer, 1996.

[6] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61:175–198, 1988.

[7] P. Eades and J. Marks. Graph drawing contest report. R. Tamassia and I. G. Tollis, editors. *Proc. 2nd Intl. Symp. Graph Drawing (GD '94)*, *Lecture Notes in Computer Science* vol. 894, pages 143–146. Springer, 1995.

[8] P. Eades and J. Marks. Graph-drawing contest report. F. J. Brandenburg, editor. *Proc. 3rd Intl. Symp. Graph Drawing (GD '95)*, *Lecture Notes in Computer Science* vol. 1027, pages 224–233. Springer, 1996.

[9] P. Eades, J. Marks, P. Mutzel, and S. C. North. Graph drawing contest report. S. H. Whitesides, editor. *Proc. 6th Intl. Symp. Graph Drawing (GD '98)*, *Lecture Notes in Computer Science* vol. 1547, pages 423–435. Springer, 1998.

[10] P. Eades, J. Marks, and S. C. North. Graph-drawing contest report. In S. C. North, editor, *Proc. 4th Intl. Symp. on Graph Drawing (GD '96)*, *Lecture Notes in Computer Science* vol. 1190, pages 129–138. Springer, 1996.

[11] P. Eades, J. Marks, and S. C. North. Graph-drawing contest report. G. Di Battista, editor. *Proc. 5th Intl. Symp. Graph Drawing (GD '97)*, *Lecture Notes in Computer Science* vol. 1353, pages 438–445. Springer, 1997.

[12] K. Mehlhorn and S. Näher. *The* LEDA *Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999. Project home page `http://www.mpi-sb.mpg.de/LEDA/`.

[13] K. Sugiyama and K. Misue. Graph drawing by the magnetic spring model. *Journal on Visual Languages and Computing*, 6(3):217–231, 1995.

[14] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, 1981.

[15] R. Webber. *Finding the Best Viewpoint for Three-Dimensional Graph Drawings*. PhD thesis, University of Newcastle, 1998. See `http://www.cs.mu.oz.au/~rwebber/research/thesis/`.