Fast Interactive 3-D Graph Visualization

Ingo Bruß*, Arne Frick**

Universität Karlsruhe, Fakultät für Informatik, D-76128 Karlsruhe, Germany

Abstract. We present a 3-D version of GEM [6], a randomized adaptive layout algorithm for nicely drawing undirected graphs, based on the spring-embedder paradigm [4]. The new version, GEM-3D, contains several improvements besides the adaptation to 3-D geometry.
The main result of this work is that for the first time, 3-D layout and presentation techniques are combined available at interactive speed. Even large real-life graphs with hundreds of vertices can be meaningfully displayed by enhancing the presentation with additional visual clues (color, perspective and light) and the possibility of interactive user navigation.
In the demonstration, we interactively visualize many graphs (artificial and real-world) of different size and complexity to support our claims. We show that GEM-3D is capable of producing a textbook-like drawing of the PETERSEN graph, a notoriously hard case for automatic drawing tools. To the best of our knowledge, this has not been achieved before by automatic layout algorithms purely based on heuristics.

1 Introduction

Visualizations of large discrete data structures are becoming increasingly more important in the literature [11, 16, 17] due to their practical relevance. Many discrete data structures from the real world can be modeled as (large) graphs, thus creating a need for automatic layout strategies to display them.

This paper introduces a new 3-D graph layout algorithm based on the well-known *spring-embedder* approach [4, 7, 9]. Spring-embedder algorithms have the ability to produce graph embeddings in the plane that look like projections of 3-D layouts onto a drawing area. It is therefore natural to extend a spring-embedder algorithm to 3-D and explore the effects. We chose the GEM (graph *em*bedder) algorithm [6] for this study due to its excellent performance in terms of both layout time and the quality of the resulting drawings. In addition, GEM scales well to graphs with several hundred vertices, thus making it quite successful in practice. Despite being randomized, GEM turned out to deliver very stable results. The GEM algorithm combines the spring-embedder approach with ideas from simulated annealing by assigning each vertex a *local temperature*.

The remainder of this paper is organized as follows. As a general approach, we distinguish between the layout and presentation aspects of the discussed topics. Section 2 reviews related work on both aspects. In Sect. 3 we present the GEM-3D layout

* EMail: bruss@wbkst10.mach.uni-karlsruhe.de
** EMail: frick@informatik.uni-karlsruhe.de

algorithm and the presentation and interaction techniques used by the GEM3DDRAW system. Section 4 discusses the results. We evaluated both the algorithmic performance and the quality of the resulting drawings on the layout side. Regarding the presentation, we state observations and experiences made with the presentation techniques employed. The summary lists possible applications and directions for further research in this area.

2 Related work

Even more than in 2-D, the display of graphs in 3-D is actually a two-fold task, consisting of a *layout* phase, drawing the graph æsthetically, and a *presentation* phase, that applies viewing strategies, techniques and tools to present a meaningful view on the graph to the observer.

2.1 Layout

Previous research on 3-D graph drawing algorithms has focused on restricted kinds of drawings, e.g. tree and orthogonal drawings [2]. Recently, it has been extended to the layout of hierarchical information [15] in 3-D. The idea of drawing arbitrary undirected graphs in 3-D seems to have first appeared in [7]. This paper and the accompanying system demonstration extend those ideas and show that it is now possible to draw even large undirected graphs with several hundred vertices and possibly thousands of edges in 3-D with interactive speed.[1] It extends a similar result for computing drawings of undirected graphs in 2-D, that confirmed an earlier conjecture in [7] by using an improved adaptive cooling schedule[6]. In the remainder of this section, we briefly recall the basic working of the underlying class of algorithms, called *spring-embedder* algorithms, before we review GEM, the predecessor to the algorithm presented in this paper.

Spring-embedder algorithms use a physical model based forces that are exerted on the vertices in order to improve their positions according to several æsthetics [4]. Once the vertices are placed, the edges are drawn as straight lines between the vertices. The model states that vertices repel each other, while adjacent vertices are attracted to each other. These simple rules define a dynamic system that can be driven into a local energy minimum. The easiest strategy to do so is to use the *gradient descent* method, according to which only downhill moves are allowed, until no further improvements are possible [9]. Other strategies of achieving convergence are the use of simple cooling schedules that restrict the allowed moves over time [7] or to apply *simulated annealing* [3, 13].

The resulting drawings satisfy a surprising number of æsthetic criteria commonly used to evaluate the quality of drawings. In particular, vertices are almost evenly spaced in the drawing area, and the deviation of the edge lengths is low. Further important criteria believed to determine the æsthetic appearance of a drawing include *edge crossings* and *symmetries*. Statistical evidence to support the former claim has recently been

[1] In this context, "interactive speed" means that the initial layout should not take less than a few seconds, perhaps more if there is some sort of progress report such as animation. In the navigation phase, the user must receive immediate response on his commands, which are mostly mouse-based.

given in [14], while the data on the latter is not yet conclusive. Two important observation on spring-embedder algorithms can be made. Although the minimization of edge crossings and the maximization is not part of their energy functions to minimize, global symmetries are often found in the resulting drawings, if they exist in the graph. Secondly, graphs are often embedded with few edge crossings, or appearing as a 2-D projection of a 3-D layout. The latter property inspired us to construct 3-D layouts using the spring-embedder model. The objective has been to achieve æsthetic layouts of general undirected graphs and their presentation with interactive speed.

As the performance of annealing-based approaches is not very promising in terms of interactive speed [3, 13], we chose to base our research on the GEM algorithm presented in [6], a randomized adaptive algorithm. Based on the spring-embedder paradigm, several key factors contribute to its outstanding performance, among them a gravitational force and several heuristics to speed up convergence. Surprisingly, the resulting algorithm is very robust under several types of randomization. For example, vertex positions are updated one at a time according to a random permutation chosen initially.

Vertex updates are grouped into *rounds*, in which each vertex is updated exactly once. The direction of a position update is computed from the force resulting from the attractive, repulsive and gravitational force, while the length of the update move depends on the last update and an adaptive component called the *local temperature* of the vertex. Local temperatures are high initially and adapt to the movements of the respective vertex. Once GEM detects an *oscillation* of a vertex (see Fig. 1a) or a *rotation* of a subgraph (see Fig. 1b) during a vertex update step, the corresponding local temperature is lowered, thus further approaching the desired energy minimum. On the other hand, if a vertex keeps going in the same direction (cf. Fig. 1c), this is interpreted as a signal to accelerate it for further moves by increasing its local temperature.

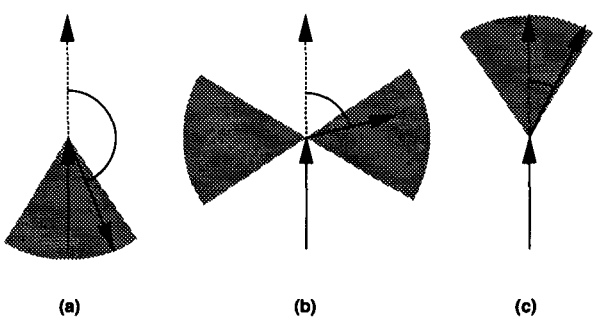

Fig. 1. Examples of vertex oscillation (a), rotation (b), and acceleration (c).

2.2 Presentation

There exist many techniques to display large graphs, e.g.

1. All information associated with the vertices and edges of the graph is displayed. The obvious drawback is that the details quickly become too small.
2. All information is drawn into a virtual drawing area, only part of which is visible at any given time. The user may browse the drawing by scrolling and arc traversal. This approach, however, tends to obscure the global structure of the underlying graph [8].
3. Several views are displayed at once: a global overview map preserving the overall structure, and one or more zoom-in maps displaying local information with details not present in the overview.
4. Distorting views such as the *fish-eye lens* approach [17] are useful to preserve the overall structure. The area within a *focal region* will be visible in detail, while other areas will be displayed distorted, i.e. smaller or with less detail. The rate of distortion rises with the distance from the focal region.

Another kind of distorting view is defined by arranging the objects under display in 3-D space, and to use a perspective projection onto the view plane. This leaves the details closest to the view plane unchanged, while objects further away are displayed smaller according to their distance to the view plane. We believe that 3-D views are more natural and therefore more intuitive than fisheye views, because they are closer to the human visual perception, although statistical evidence to support this belief is still missing.

There are several techniques to display 3-D pictures on a 2-D computer screen, among which *animation* is probably the simplest. Based on a single picture, the illusion of depth is created by having pictures differ slightly over time. *Stereopsis* requires two different pictures at any time, one for each eye. It may be realized by e.g. polarizing filters or holography. For simplicity reasons, we based the presentation of our layout results only on animation, although nothing prohibits the use of other techniques. Depth perception may be supported by graphical depth cues such as *color* and *light* to improve user understanding. Interaction techniques, such as *rotation*, *zoom*, *translation* of the structure as a whole, and the *selection* of artifacts within the structure further contribute [5, 16] to this objective.

3 The GEM3DDRAW system

3.1 Layout

Dependent parameters In contrast to previously known spring-embedder algorithms, GEM-3D allows for vertices of different *shape* and *extent*. This information is used in the layout algorithm to determine a lower bound for the *minimum vertex separation*

$$V_{\text{min}} = v_{\text{avg\_size}},$$

which is given by the average vertex size $v_{\text{avg\_size}}$. The minimum vertex separation also influences another parameter, the so-called *optimal* or *desired* edge length, previously defined as a constant that would usually depend on the size of the drawing area [9]. In GEM-3D, the desired edge length is computed as

$$E_{\text{des}} = v_{\text{avg\_size}} \cdot \overline{v.deg},$$

where $\overline{v.deg}$ is the average edge degree of the vertex set. This serves the purpose of avoiding tightly packed layouts for highly connected graphs.

We shall now describe the GEM-3D algorithm in more detail. In doing so, we shall especially mention the differences regarding the 2-D version. The transformation of GEM from 2-D to 3-D was mostly straight forward, as the algorithm contains nothing inherently two-dimensional. Of course, the computation of Euclidean distances had to be adapted. The notion of opening angles was extended to opening cones. The most difficult part was the adaptation of the convergence-speedup heuristics. Figure 2 shows the algorithm at an abstract level. As convergence to a local minimum has not yet been formally proven, we ensure termination by using the constant $R_{\max}$ as an "emergency exit".

```
-- Input:                                                              1
--     G = (V, E) graph where                                          2
--     V = set of record                                               3
--         s  -- shape type and size information                       4
--         ξ  -- current position                                      5
--         p  -- last impulse                                          6
--         t  -- local temperature                                     7
--         d  -- skewness gauge (see text)                             8
--     E = set of record                                               9
--         v1 -- adjacent vertex 1                                     10
--         v2 -- adjacent vertex 2                                     11
--         d  -- [optional:] directedness flag                         12
--     Rmax   maximal number of rounds                                 13
-- Output: for each v ∈ V, a position is computed                      14
                                                                       15
compute Tmin, the desired minimal temperature                          16
compute Edes, the optimal edge length                                  17
compute Emin, the minimal edge length                                  18
forall v ∈ V do                                                        19
    initialize v;                                                      20
nrounds := 0;                                                          21
while Σi v.t > Tmin and nrounds< Rmax do                               22
    forall v ∈ V do                                                    23
        compute v.p;                                                   24
        update v.ξ and v.t;                                            25
    nrounds := nrounds+1;                                              26
```

Fig. 2. Main loop of the GEM-3D algorithm.

Initialization and vertex choice remain the same as in the 2-D version. Vertices may be positioned randomly initially, and vertices are chosen for updates according to a random permutation.

Impulse computation We consider the computation of a new impulse for a given vertex $\mathbf{u}$. The attractive, repulsive and gravitational forces $\mathbf{F_a}, \mathbf{F_r}$ and $\mathbf{F_g}$ are added up and smeared a little by adding a Brownian motion component consisting of a small random vector ρ with expectation $\mathbf{0}$:[2]

$$\mathbf{p} = \sum_{\mathbf{v} \in V} \mathbf{F_r}(\mathbf{u}, \mathbf{v}) + \sum_{(u,v) \in E} \mathbf{F_a}(\mathbf{u}, \mathbf{v}) + \mathbf{F_g}(\mathbf{u}) + \rho, \tag{1}$$

resulting in the *current impulse* of the vertex, that is further scaled by the local temperature $\mathbf{v}.t$ and then applied to $\mathbf{v}$ as movement relative to its current position.[3]

Position and temperature update Figure 3 shows the abstract algorithm for updating the position and temperature of a vertex v. The position of v is updated by the current impulse, which is scaled with the current temperature of v. Afterwards, the temperature of a vertex is updated according to the rotation and oscillation detection heuristics. As shown in Fig. 1, a vertex is accelerated (decelerated) if it moves into the same (opposite) direction twice in a row, which can be detected by storing the impulse computed at the last update. As in the 2-D case, the definition of "same" and "opposite" depend on the opening angle α_o. The influence of oscillation detection is determined by a constant $\sigma_o < 1$.

Unfortunately, the definition of "rotation" cannot be naturally extended from 2-D to 3-D. In GEM, there is a skewness gauge indicating trends towards rotations by adding 1 for each approximate right angle movement to the left and -1 for each approximate right angle movement to the right. If the absolute value of the resulting counter becomes larger than a threshold τ, then the local temperature is reduced. The objective is to dampen vertex movements in situations such as rotating subgraphs.

The extension of this mechanism would obviously require an infinite number of skewness gauges, as there are infinitely many planes of rotation. Therefore, three alternatives were considered.

1. three counters, one for each of the coordinate planes $x-y$, $x-z$, and $y-z$. The counters would be used to detect rotations in the projections of the last and current impulse vector onto the coordinate planes.
2. a single counter to count the number of approximate 90° angles.
3. a global cooling schedule instead of rotation detection. In fact, a rotation detection scheme with large opening angle σ_r and small sensitivity σ_r is very similar to a cooling schedule, because it will almost always fire.

[2] The gravitational force draws vertices towards the barycenter of the current layout in order to keep disconnected components from drifting apart. However, we found that the constants should be lower than in the 2-D case. Otherwise, the graph will be embedded too tight into the drawing space. In addition, a low gravitational force effects smoother embeddings than a higher.

[3] In equation 1, the summation goes over all vertices for the repulsive force and all adjacent vertices for the attractive force.

```
-- Input:                                                    1
--   v        vertex to be updated                           2
--   p        current impulse of v                           3
-- Output:                                                   4
--   v        with updated ξ, t, d, p                        5
                                                            6
scale p by v.t                                              7
v.ξ := v.ξ + p;                                             8
β:=∠p, v.p;                                                 9
case β of                                                   10
    Acceleration then Increase v.t                          11
    Oscillation then Decrease v.t                           12
    Rotation then small decrease of v.t                     13
esac                                                        14
v.p:=p;                                                     15
```

Fig. 3. Position and temperature update algorithm.

3.2 Presentation

User support for *navigation* is available by user-controlled rotation, translation and zooming. In addition, a vertex may be selected by its name, using a pop-up menu or an entry widget in the user interface. The selected vertex will subsequently be rotated to the front of the display to also show its context.

In order to support *depth perception* of a 3-D image from a 2-D picture, we used several of the techniques mentioned in Sect 2.2. *Perspective* contributes to the perception of depth in the drawing by changing the size of vertices depending on their distance from the view plane. *Colors* can effectively help to reveal structure. In addition, a new technique called *artificial fog* was introduced. It modulates the color of objects depending on their distance from the view plane, e.g. by adding more and more of a dark color to make distant objects not only smaller, but also darker.

4 Results

In this section we summarize the results achieved so far with our prototype of GEM-3D. Again, algorithmic results are distinguished from results based on the presentation techniques. For obvious reasons, the pictures shown below cannot convey the same information as a self-guided, animated exploration of the graphs. Therefore, we have made our implementation publicly available for major platforms.[4]

[4] URL: `http://i44www.info.uni-karlsruhe.de/~frick`.

4.1 Layout

In this section, we consider both the quality and the performance of the layout algorithm.

Quality As for the quality of the resulting layouts, we observed that

- Like all spring-embedder algorithms, GEM-3D is good at separating vertices as much as possible in the drawing space (see Fig. 4 and 6).
- Vertices do not touch each other in general. No overlaps have been encountered yet.
- Although the focus of the algorithm is on fast convergence, symmetries are still often found (see Fig. 4 and 7).
- The 3-D topology of 3-D graphs is displayed (see Fig. 4 and Fig. 5).
- A textbook-like embedding of the PETERSEN graph can be produced (see Fig. 8 and Fig. 9) with an observed chance of 1 in 3–4 tries.

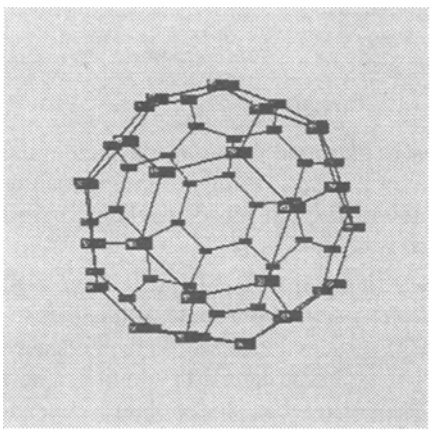

Fig. 4. The buckminsterfullerene molecule C_{60}. These examples shows the 3-D display, symmetry detection and vertex separation properties of GEM-3D.

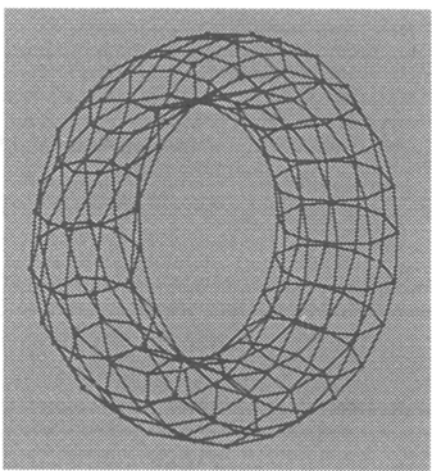

Fig. 5. Torus with $|V| = 200, |E| = 400$. This figure took 6 seconds to be generated.

Performance All measurements were done on an SGI Indy workstation with a MIPS R4600 processor in normal daily use. Whenever times are given, we shall also provide the number of rounds necessary to compute the respective layout, as the inner loop computing the forces depends only on the graph size, while the outer loop appears to convey the complexity of the graph in terms of the number of rounds necessary to converge.

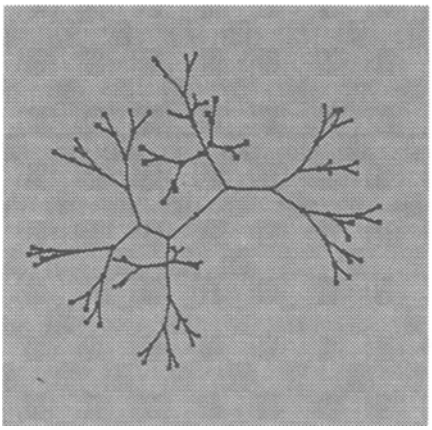

Fig. 6. Complete binary tree of height 7. This is another example of the vertex separation property.

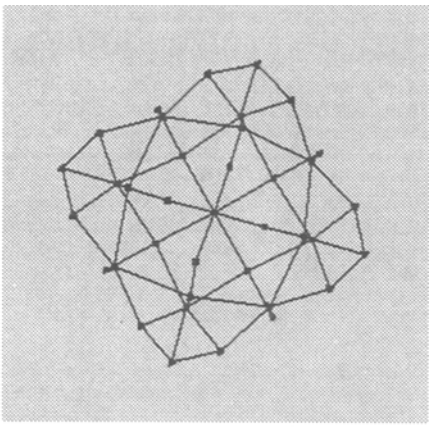

Fig. 7. The highly symmetric graph in Fig. 26 of [7]. Although appearing to be a 2-D drawing, the third coordinate dimension is used to separate vertices, as the color differences indicate.

It turns out that the larger the graph gets, the faster is GEM-3D over its predecessor: The layout of the graph in Fig. 10 took only 65 rounds on the average, using 5 seconds of user time, an 80% increase over GEM, which required more than 100 rounds on the average. The 20x10 torus from Fig. 5 required a total user time of 6 seconds for the layout and then another 2 seconds to bring up the view. We determined empirically that the number of rounds required for convergence is usually sublinear in the number of vertices. For the worst case, the choice of $R_{\max} = 5$ guarantees a linear number of rounds, resulting in a total complexity of $O(|V|^3)$ vertex updates. Even in this rarely occuring worst case, the resulting drawings are usually quite acceptable.

Rotation and oscillation detection both continue to contribute to the overall performance. The continuously good performance over a wide range of parameters and heavy use of randomization renders GEM-3D a very good candidate as a 3-D layout algorithm for undirected graphs.

4.2 Presentation

Experiments with GEM-3D support the initial hypothesis that depth cues help to convey much information in little screen space. On the other hand, graphs like the $K_{3,3}$ cannot be recognized by non-expert users without a coloring of the vertices hinting at the partition (see the right side of Fig. 11). This is an example of a more general problem with displaying information as graphs: Edges only convey neighborhood information, but sometimes the data structure contains information not being displayed by edges.

The selection mechanism proved to be useful to instantly focus on selected vertices, which may be hidden inside a large conglomerate of vertices and edges.

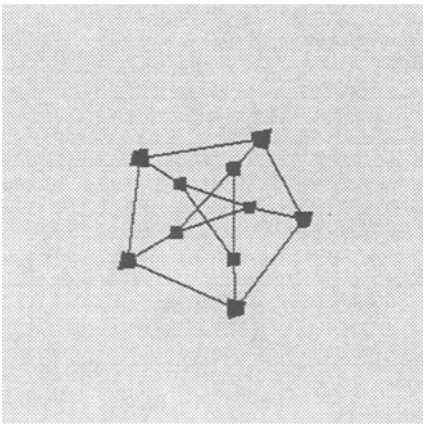

Fig. 8. Text-book drawing of the PETERSEN graph produced by GEM-3D. There is a fairly good chance to find such a drawing.

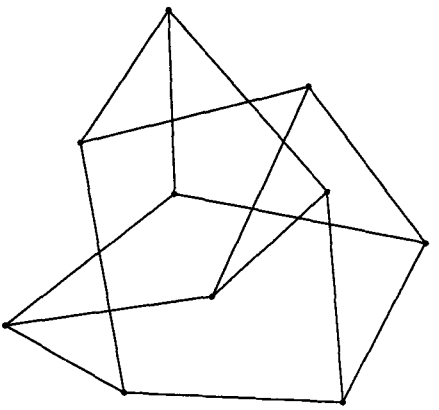

Fig. 9. Typical drawing produced by 2-D spring-embedder algorithms.

5 Conclusions

In this paper, we have shown how to extend the GEM algorithm to 3-D coordinate dimensions while significantly improving the layout speed and introducing new presentation techniques. The overall objective of drawing and presenting large graphs with interactive speed has been achieved.

We applied GEM-3D to real-world graphs, as well as to artificial ones. The results show that the algorithm draws 3-D structures as such, providing the viewer with helpful insights into their topology. For large, not too dense graphs, the resulting drawings are of high quality. In this case, the algorithm positions highly interconnected connected vertices into well-separated clusters. This is especially useful if combined with user interaction and depth cues (color and size of vertices).

3-D drawings may help to reveal structure in graphs unaccessible to known automatic 2-D graph layout algorithms. A good example, that may also be of some theoretical interest, is the fact that GEM-3D is able to produce a textbook-like drawing of the PETERSEN graph. Until now, it was impossible to achieve this result using spring-embedder algorithms, probably due to the fact that the textbook drawing violates the æsthetic criteria coded into the forces (spacing of vertices, minimization of edge length deviation).

Due to its potential to visualize even large structures in a very intuitive way, GEM-3D is now used in CAKETOOL, a large AI software system, to visualize Bayesian networks [1]. This provides evidence that the time has come for interactive 3-D graph visualization combined with automatically generated layouts. Other possible applications include the display of social networks [12] and the use as component in multimedia applications for displaying large hypertext-like structures.

Depth cues will be even more helpful if supported by light sources placed outside

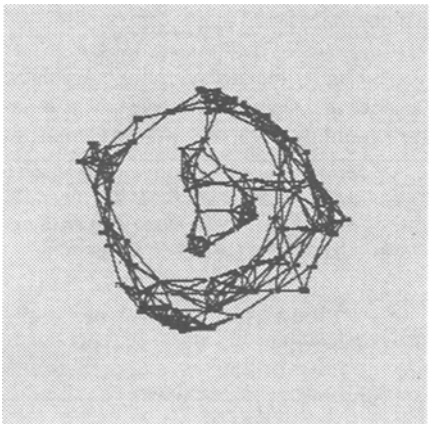

Fig. 10. Example of a large real-world graph ($|V| = 128, |E| = 508$), defined in [10] as *miles(128,0,0,0,0,10,4711)*.

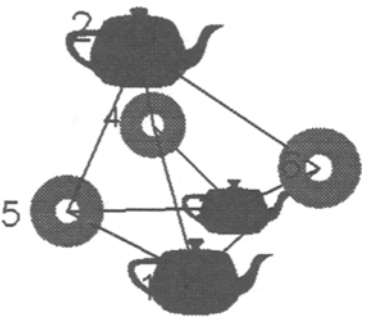

Fig. 11. Example of how color, vertex shapes and labels help to reveal the structure of a graph, in this case a $K_{3,3}$. Vertices are colored, shaped and labeled according to their layer.

the drawing space. Experiments to this end are promising. Another promising approach is to substitute the rotation detection part by a simple global cooling schedule, as outlined in Sect. 3.1.

The current version of GEM-3D can draw directed graphs by displaying appropriate arrows on the edges. However, this does not imply that the forces between the connected vertices are treated as unidirectional. Therefore, the layout presently remains the same for the directed and undirected version of the graph. In the future, we plan to study the effect of one-way forces exerted by directed edges. Of course, this will lead us further away from the original physical interpretation of the state equations involved.

GEM3DDRAW was designed to be extensible. This allowed for the addition of further node shapes and visualization techniques (fog, perspective). Further research will explore the use of constraints to accommodate other æsthetic criteria and user knowledge about the data structure.

In conclusion, the GEM family of algorithms remains an attractive choice for the practitioner, while at the same time posing a challenge to theoreticians to actually *prove* the run-time and convergence behavior we observe so consistently, even under heavy randomization.

References

1. Ingo W. Bruß. Konzeption und Realisierung einer Visualisierungskomponente für komplexe Datenstrukturen unter dem Werkzeug CAKETool. Master's thesis, Universität Karlsruhe, 1995.

2. R. F. Cohen, P. Eades, T. Lin, and F. Ruskey. Three–dimensional graph drawing. In *Proceedings of Graph Drawing'94*, volume 894 of *LNCS*, pages 1–11. Springer, 1994.
3. R. Davidson and David Harel. Drawing graphs nicely using simulated annealing. Technical Report CS89-13, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, Israel, 1989. revised July 1993, to appear in Communications of the ACM.
4. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.
5. Kim M. Fairchild, Steven E. Poltrock, and George W. Furnas. *SemNet: Three-Dimensional Graphic Representations of Large Knowledge Bases*, chapter 5. Lawrence Erlbaum associates, 1988.
6. Arne K. Frick, Heiko Mehldau, and Andreas Ludwig. A fast adaptive layout algorithm for undirected graphs. In *Proceedings of Graph Drawing'94*, volume 894 of *LNCS*, pages 388–403. Springer, 1994.
7. T.M.J. Fruchterman and E.M. Reingold. Graph drawing by force-directed placement. *Software–Practice and Experience*, 21, 1991.
8. J.G. Hollands, T.T. Carey, M.L. Matthews, and C.A. McCann. Presenting a graphical network: A comparison of performance using fisheye and scrolling views. In *Proceedings of the 3rd International Conference on Human-Computer Interaction*, pages 313–320, September 1989.
9. T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31, 1989.
10. Donald E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. ACM Press, New York, 1993.
11. J.D. Mackinlay, George G. Robertson, and S.K. Card. The perspective wall: Detail and context smoothly integrated. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 173–179. ACM, 1991.
12. Cathleen McGrath, Jim Blythe, and David Krackhardt. The effect of graph layout on inference from social network data. In *Proceedings of GD'95*, 1995.
13. Burkhard Monien, Friedhelm Ramme, and Helmut Salmen. A parallel simulated annealing algorithm for generating 3D layouts of undirected graphs. In *Proceedings of GD'95*, 1995.
14. Helen C. Purchase, Robert F. Cohen, and Murray I. James. Validating graph drawing æsthetics. In *Proceedings of GD'95*, 1995.
15. S.P. Reiss. 3-D Visualization of Program Information. In R. Tamassia and I. Tollis, editors, *Graph Drawing DIMACS International Workshop GD '94*, number 894 in LNCS, pages 12–24. Springer Verlag, 1994.
16. George G. Robertson, J.D. Mackinlay, and S.K. Card. Cone trees: Animated 3-D visualizations of hierarchical information. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1991.
17. Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. *Comm. of the ACM*, 37(12):73–84, December 1994.