Multi-Ani: A Interface for Animating Multiple Graph Drawing Algorithms

Mao Lin Huang
Department of Computer Systems
Faculty of Information Technology
The University of Technology, Sydney
NSW 2007, Australia.

Abstract *Multi-Ani is an interface written in Java that can be used for implementing and visualizing graph-drawing algorithms with smooth transitions between layouts.*

Multi-Ani displays a sequence of drawings, D_1, D_2, ... D_n of a graph, G = (V, E); each drawing D_i is a key frame of the visualization that is calculated by a particular drawing algorithm A_i. The system uses the layout animation to produce a sequence of "in-betweening" screens between any two drawings.

This gives the user a visual impression of how a layout algorithm works, and what the differences are between two drawing algorithms by showing a sequence of smooth, continuous interpolations. It also preserves the mental map of the user after each transformation.

Keywords: information visualization, algorithm animation, in-betweening, interpolation, graph drawing.

1 Introduction

The are many types of information visualization techniques exists. Some show the network data traffic, the flow chart of a process, and database entries manipulated by a user or program. However, Algorithm Animation only concerns the presentation of the procedural steps of a specific algorithm. The animation techniques can be used to assist in portraying the steps of an algorithm.

Throughout the history of algorithm animation, advances in computer hardware have driven the evolution of algorithm animation systems. In 1986, the system *Animus* [1] demonstrated the utility of smooth transformations of 2D images of small size graphs. In the late 1980's, the system *TANGO* [2] provided a framework for 2D algorithm animation. In the early 1990's the system *Zeus* [3] proposed "algorithm auralization" - using non-speech sound to convey the workings of algorithms [4]. Other systems, such as *GraphVBT* [5], can also be used for programming algorithm animations.

This paper describes an interface, *Multi-Ani* written in Java that can be used for implementing and animating graph drawing algorithms. *Multi-Ani* provides visual representations for vertices, edges and other visual attributes that are associated with vertices and edges. It uses the layout animation [6,7], to smooth the transitions between key frames.

Another problem with the transitions between two drawing algorithms is the "mental map" problem. When the drawing jumps from one algorithm to another, there is no smooth transformation between the layouts. The user's mental map of the view is broken, and thus the user has to spend extra cognitive effort to re-form the mental map after each transition. In *Multi-Ani* we use layout animation to guide the user between views: they make the transitions naturally and smoothly. In the user's visual sense, there is only one animated image changing from one algorithm to another. This greatly reduces the cognitive effort in re-forming the user's mental map after each transformation.

2 A graph model with multiple drawings

A *graph* consists of a finite set V of nodes and a finite set E of edges, where each edge is an unordered pair of nodes of graph G. A node μ is said to be *adjacent* to a node v if (μ, v) is an edge of G; in this case, the edge (μ, v) is said to be incident with μ and v.

A *straight line* drawing of a graph $G = (V, E,)$ is a function $D: N \rightarrow R^2$ that associates a drawing $D(v)$ to each node $v \in N$. Since all drawings in this paper are straight line drawing we omit the term "straight line".

A drawing D of a graph $G = (V, E)$ consists of a location for each node $v \in V$ and a route for each edge $e \in E$.

A set of different drawing algorithms $A_1, A_2, \ldots A_n$ can produce a sequence of different drawings $D_1, D_2, \ldots D_n$ of the graph G; each drawing D_i is a *key frame* of the visualization calculated by drawing algorithm A_i.

In the actual layout creation, a drawing $D(v)$ of a node $v \in N$ is normally represented by a graphic box (perhaps enclosing some text) appearing on the screen with the position (x_w, y_v) at the center of the box, where (x_w, y_v) are the pixel coordinates of a reference point of the node. Therefore, there are two additional graphic attributes h_v and w_v associated with each drawing $D(v)$, where h_v represents the height of the graphic node and w_v represents the width of the graphic node.

3 The Animation Model and the method for calculating Key frames

In this section, we present the details of our visualization technique. We describe the animation algorithm that we use to provide a sequence of animated drawings that transform smoothly from one layout algorithm to another. These drawings also assist the user in preserving their mental map of the view as they move from one graph drawing algorithm to another. We discuss the layout adjustment method that we used to solve the overlap problem. We present the layout animation mechanism that guarantees the smooth transitions between drawings.

We now describe the animation model. We use force-directed animation algorithm [6,7] to produce key frame sequences. It is the combination of *Hooke's* law spring and *Newtonian* gravitational forces.

Key frame sequences occur in many interactive systems [2, 5], that handle relational information. Most such systems suffer from the ``mental map" problem: a small logical change in the graph results in a large change in relative positions of nodes in the drawing. The transition between key frames is smoothed by ``in-betweening". This technique aims to achieve the twin goals of good layout and the preservation of the mental map.

The in-betweening consists of a sequence D_{i+1}^0, D_{i+1}^1, D_{i+1}^2, ..., $D_{i+1}^k = D_{i+1}$ of drawings of G called *screens*. In *Multi-Ani*, screens are computed by using a force-directed algorithm, described in [spring]. The locations of the nodes in D_{i+1}^j differ only slightly from the locations of the nodes in D_{i+1}^{j+1}; the appearance is that of all the nodes moving slowly.

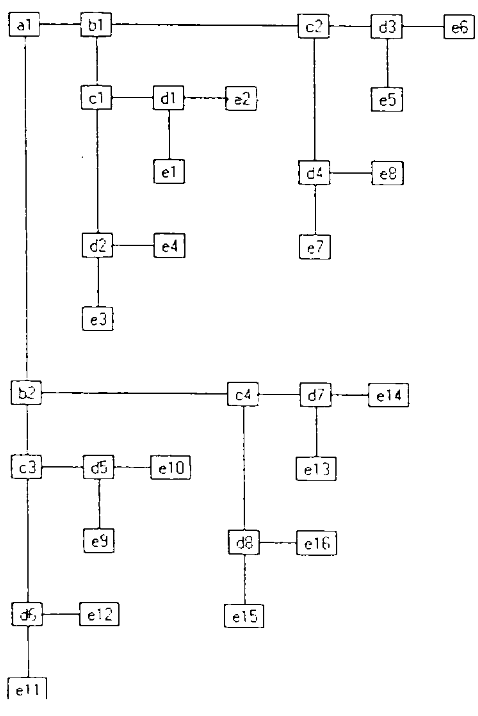

Figure 1: An initial key frame D_0 of a graph G applied by the h-v drawing algorithm.

When a new layout algorithm A_{i+1} applied onto the graph G, the old drawing D_i moves slowly toward the new drawing D_{i+1}. All the nodes in D_i move from their old positions to the new positions according to the spring and gravitational forces: each screen $D_{i+1}{}^{j}$ has energy a little lower than that of the last screen $D_{i+1}{}^{j-1}$.

An *equilibrium drawing* of the G is a drawing in which the total force $f(v)$ on each node $v \in G$ is zero. Equivalently, the model seeks to find a drawing in which the potential energy is locally minimal with respect to the node positions. The key frame D_{i+1} is at equilibrium.

Consider the in-betweening sequence $D_{i+1}{}^{0}$, $D_{i+1}{}^{1}$, $D_{i+1}{}^{2}$, ... $D_{i+1}{}^{k} = D_{i+1}$ of screens leading from the key frame D_i to the key frame D_{i+1}. This begins with a drawing $D_{i+1}{}^{0}$ that is not at equilibrium, but as it differs very little from D_i, it is close to equilibrium. Each $D_{i+1}{}^{j}$ is a little closer to equilibrium; that is, the animation is driven by the a drawing algorithm moving the nodes toward equilibrium positions.

This is accomplished by moving each node v a small amount proportional to the magnitude of $f(v)$ in the direction of $f(v)$ at each step.

We use a very simple numerical technique to minimize energy. The technique has two aims. The first is to find an equilibrium layout. The second is to produce smooth motion on the screen, that is, to give a visual effect of continuous movement. The second aim implies that we do not use a complex (and perhaps faster) numerical technique because it may produce a jerky motion.

4 A example of a smooth transition between two drawing algorithms

To illustrate how the system works, a simple example session is presented in this section.

The system first displays an initial key frame D_0, a *h-v drawing* of the graph G which is based on the h-v drawing algorithms [8,9]. The drawing starts to move from key frame D_0 *to* D_1, a spring drawing, when a force-directed algorithm applied to the G. The system applies a layout animation to produce a sequence of in-betweening screens, see Figures 1 to 8. These screens provide a smooth transition from a *h-v* layout to a *spring* layout.

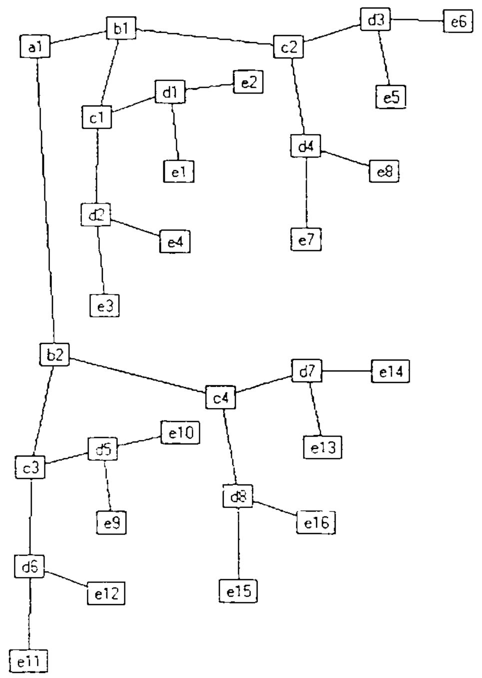

Figure 2: An in-betweening screen $D_1{}^{a}$ that moves towards the final drawing D_1.

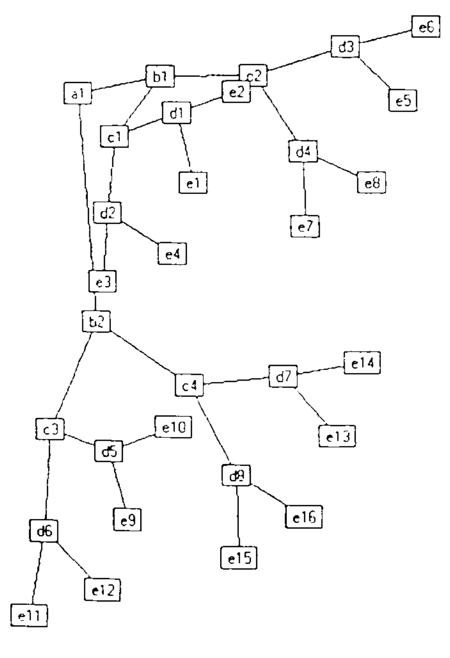

Figure 3: An in-betweening screen D_I^b that moves towards the final drawing D_I.

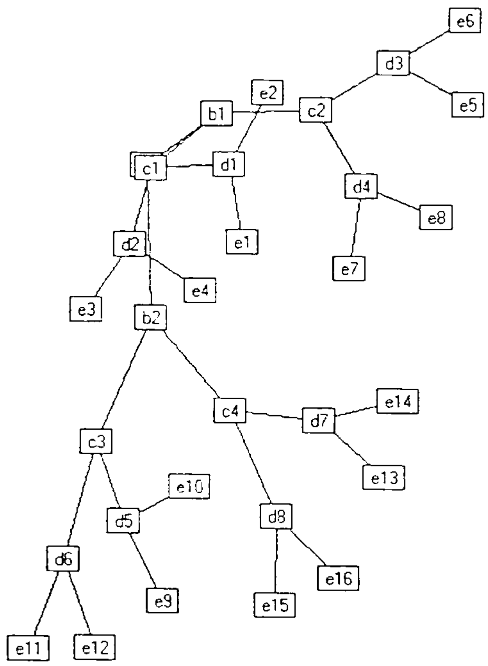

Figure 4: An in-betweening screen D_I^d that moves towards the final drawing D_I

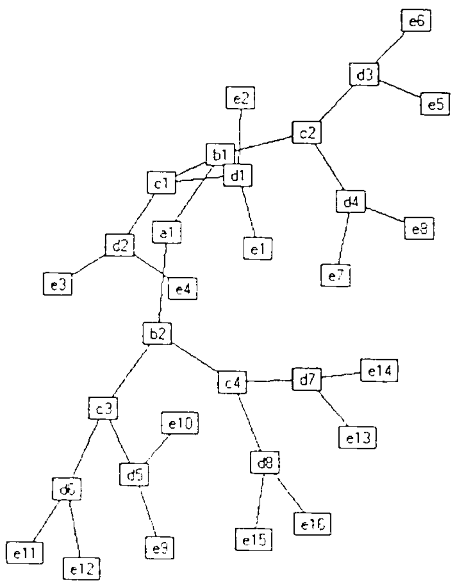

Figure 5: An in-betweening screen D_I^c that moves towards the final drawing D_I

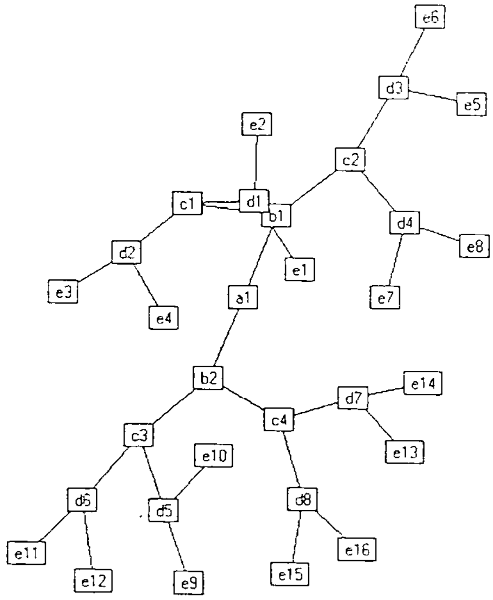

Figure 6: An in-betweening screen D_I^f that moves towards the final drawing D_I

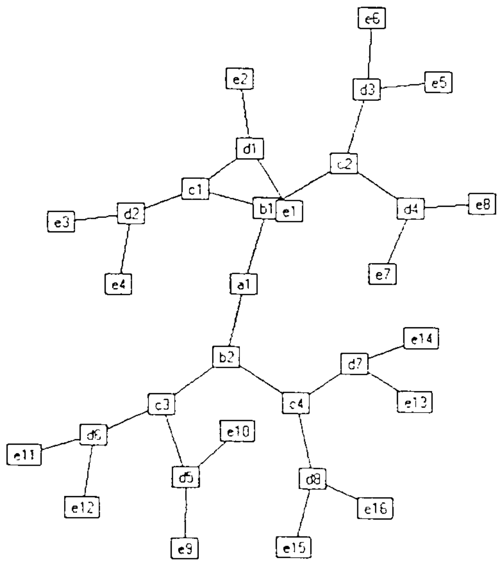

Figure 7: An in-betweening screen D_I^g that moves towards the final drawing D_I

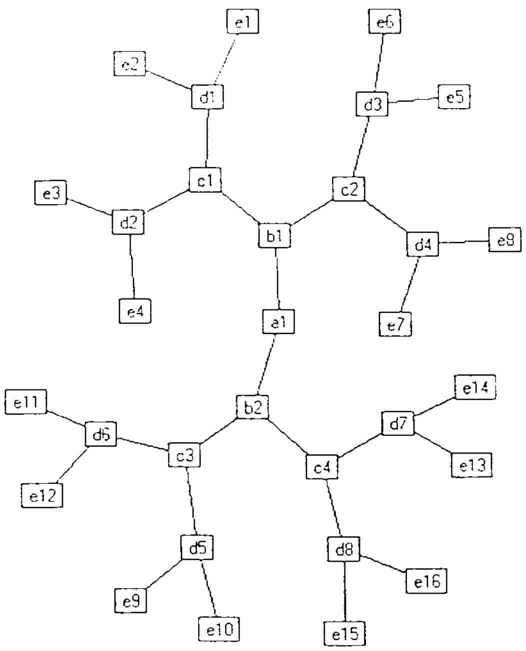

Figure 8: The final screen D_l^h, which is a new key frame reaching a spring drawing D_l applied by a force-directed algorithm.

5 Conclusion

By showing the steps of a graph drawing algorithm, continuous animation can be a useful tool to help viewers follow the operations that are occurring. Animation helps viewers identify and track changes between states, thus helping them understand how the operations evolve over time.

This paper has described the use of a interface *Multi-Ani* for animating a sequence of drawings with different drawing algorithms. This helps viewers to identify the differences and track the change of layouts between drawing algorithms, as an addition to the normal algorithm animation.

References

[1] R. Duisberg, *Animated Graphical Interface Using Temporal Constraints*, Proceedings of the ACM SIGCHI'86 Conference on Human Factors in Computing Systems, Boston, MA, April 1986, pp. 131-136.

[2] J. Stasko, *TANGO: A Framework and System for Algorithm Animation*, Computer, vol. 23, No.9, September 1990, pp. 27-39.

[3] M. Brown, *ZEUS: A System for algorithm Animation and Multi-view Editing*, Proceedings of the 1991 IEEE Workshop on Visual Languages, Kobe, Japan, October 1991, pp. 4-9.

[4] M. Brown and J. Hershberger, *Color and Sound in Algorithm Animation*, Computer, vol.25, no.12, December 1992, pp. 52-63.

[5] J. DeTreville, (1993), *The GraphVBT Interface for Programming Algorithm Animations*, in 'Proceedings of the 1993 IEEE Symposium on Visual Languages, Bergen, Norway', pp. 26-31.

[6] P. Eades, *A Heuristic for Graph Drawing* Congressus Numerantium, vol 42, pp. 149–160, 1984.

[7] M.L. Huang, P. Eades, and J. Wang, *Online animated graph drawing using a Modified Spring algorithm*, in: Proc. of the 21st Australasian Computer Science Conference (ACSC'98), 1998, pp. 17–28.

[8] P. Eades, T. Lin and X. LIN, *Minimum Size H-V Drawings*, Advanced Visual Interfaces 1992, Rome, Italy, World Scientific Series in Computer Science Volume 36, 386-394, 1992.

[9] P. Eades, X. LIN and R. Tamassia, *A New Approach for Drawing a Hierarchical Graph*, Proc. of Second Canadian Conference on Computational Geometry, 142-146, 1990.