

Building Virtual Worlds with the The Big-Bang Model

Neville Churcher and Alan Creek

Software Visualisation Group, Department of Computer Science,
University of Canterbury, Private Bag 4800,
Christchurch, New Zealand.
`neville@cosc.canterbury.ac.nz`

Abstract

Visualisations implemented as virtual worlds can allow users to comprehend large graphs more effectively. Good 3D layout algorithms are an important element. ANGLE has been developed as a platform for experimenting with 3D force-directed layout algorithms. The big-bang modification is proposed as a means of obtaining efficiently good 3D layouts for a wide range of graphs. Results are presented and compared with those from a conventional approach.

1 Introduction

Graphs are a fundamental tool for conveying information about abstract structures and empirical data. They are the basis for diagramming techniques such as UML. As the size and complexity of graphs increase people find it much harder to comprehend them. Large graphs arise frequently in our work on the visualisation of software structure, software reuse and web site activity [4, 9] and our challenge is to find ways of presenting them more effectively. Good physical layout aids comprehension but invariably involves conflicting criteria so we require widely-applicable robust techniques.

We live in a 3D world and there is strong empirical evidence that the use of 3D presentation techniques enables people to comprehend graphs 2–3 times larger than if only 2D presentation is available [12]. We have been experimenting with non-immersive virtual reality (VR) as a way to provide users with a more natural experience of graph visualisation tasks. The use of VR techniques means that choice of viewpoint and projection are of little direct relevance, since they are under user control, and the number of edge crossings is dramatically reduced.

We use the Virtual Reality Modelling Language (VRML) [3] to present our layouts. The appealing features of VRML include its simple text format which is amenable to generation and processing by software tools. Figure 1 shows a VRML browser implemented as a Netscape plug-in. Such browsers are available for many platforms. A range of controls is provided to allow navigation through and manipulation of the world.

We are not primarily interested in achieving optimal drawings of well known graphs. Rather, we wish to be confident that we can achieve a “good enough” drawing of any graph arising in the context of our user-driven just-in-time visualisation research.

In this paper we report our applications of force-directed layout techniques to the generation of 3D virtual worlds suitable for our visualisation projects. A modification introducing a “big-bang” phase is presented and an application, ANGLE we developed to experiment with algorithm modifications is discussed.

The remainder of the paper is structured as follows. A brief overview of force-directed methods is given in the next section followed by some observations on controlling the algorithms. ANGLE is described in Section 4 and the big bang model inspired by our experience with it is introduced in Section 5. Some results are presented in Section 6 and our conclusions and plans for further work appear in Section 7.

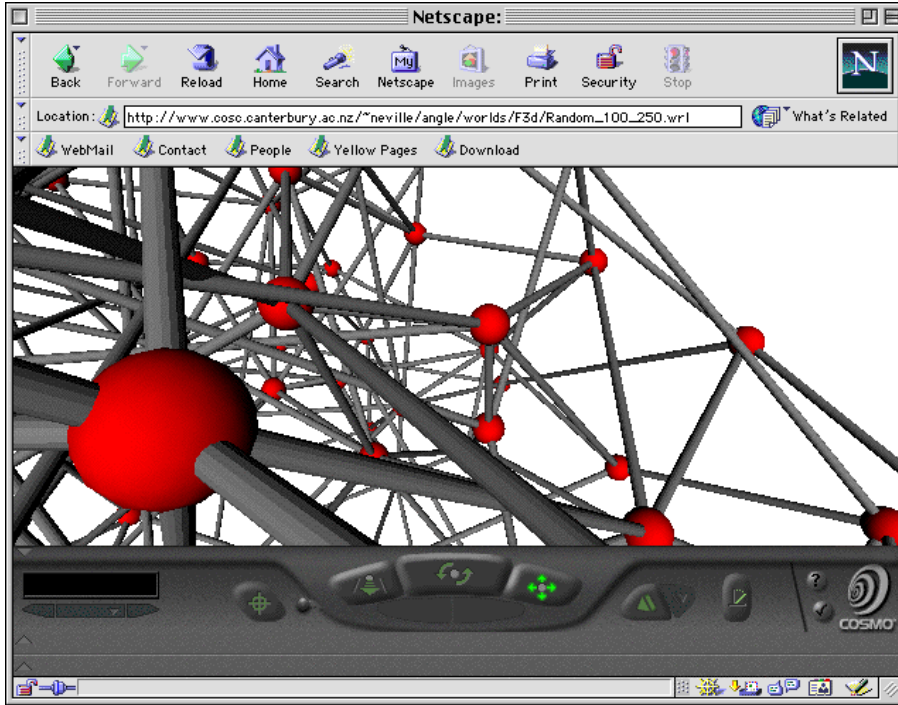


Figure 1: Exploring a virtual world containing a random graph with 100 nodes and 250 edges

2 Force-directed layout

The basic idea is to regard an undirected graph $G = (V, E)$ as a system of interacting nodes which exert pairwise repulsive forces, \mathbf{F}_{ij}^r , on each other countered by attractive forces, \mathbf{F}_{ij}^a , between pairs of nodes connected by edges. The layout is determined by computing the equilibrium positions of the nodes. The nett force on a node ν_i is then given by

$$\mathbf{F}_i = \sum_{j=1, j \neq i}^N \mathbf{F}_{ij}^r + \mathbf{F}_{ij}^a \quad (1)$$

The functional form of the attractive and repulsive forces is often chosen so as to have some relationship with familiar physical systems. One common choice is to have the attractive forces based on Hooke's law, $F = -k(x - l_0)$, where l_0 is the natural spring length, permitting the analogy of nodes as balls connected by springs. Similarly, the repulsive force is often based on the inverse square force between charged particles, $F = \frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}^2}$, so that the nodes will spread out unless restrained by the attractive edge forces.

Substituting these functional forms in equation 1, simplifying the notation for the constants, assuming unit charges and including the directions of the forces as well as their magnitudes we obtain

$$\mathbf{F}_i = \sum_{i,j \in V, i \neq j} \frac{k^r \hat{\mathbf{r}}_{ij}}{|\mathbf{r}_j - \mathbf{r}_i|^2} + \sum_{i,j \in E, i \neq j} -k^a \left((\mathbf{r}_j - \mathbf{r}_i) - \frac{l_0(\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|} \right) \quad (2)$$

Many variations have been explored. Additional forces, such as a uniform field or forces between edges, may be added in order to impose natural direction(s) on the resulting layout. Subsets of V may be identified so that additional forces may be applied to certain nodes [7, for example]. Logarithmic or r^{-1} forces may be used to weaken attractive forces, strengthen repulsive forces or

modify the balance between them [6, 8]. The force directed layout approach was pioneered by Eades [6] and has been applied in many contexts. Further details may be found in books on graph drawing [5, 11, for example] and the *Graph Drawing* conference series.

3 Termination and quality assessment

Force-directed methods are typically implemented via a simple iterative algorithm. At each iteration the forces, and hence the displacement, on each node are computed. All nodes are then updated and the next iteration proceeds. The process stops when some terminating condition is satisfied. This may be a simple limit on the number of iterations or something much more sophisticated. Many of the variations on algorithms and terminators have been summarised by Behzadi [1].

It is important for us to be able to stop the layout process when a “good enough” layout has been achieved since processing is expensive for large graphs. We also wish to avoid displaying either violent thrashing or irritating jiggling to interactive viewers.

We have experimented with three major categories of termination conditions: *Step-based* terminators use information about the current iteration—or a small sliding window including previous iterations—to determine whether to continue processing. Global properties such as tolerances may be involved. Examples include *maximum node displacement less than ϵ* and *mean node displacement has not changed by more than ϵ for the last y iterations*.

State-based terminators use properties of the entire graph. Examples include, *the total energy is lower than E* and *the mean edge length is within $n\%$ of l_0* .

Hybrid terminators combine step-based and state-based elements. A state-based contribution to a hybrid terminator might only be evaluated every n iterations because of its greater cost. Examples include *the centre of mass has moved by less than ϵ* and *total energy has not increased in the last 10 iterations and mean displacement is less than ϵ* .

Terminators inevitably perform “blind” and it is desirable to have some coarse quality measure which can effectively “eyeball” the resulting layouts. In 3D this will also help us gain understanding of pathological cases such as layouts involving co-planar nodes.

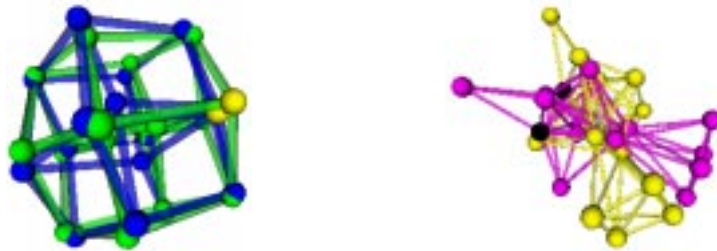


Figure 2: Qualitative assessment of layout quality *via* centre of mass alignment

The results of running an algorithm many times, differing only in initial node placement, should be a set of nearly-identical layouts. More strictly, we might expect “isomers” (chiral symmetry variations) and locally optimal layouts to be represented. By comparing the result sets for particular graphs we may hope to determine appropriate parameters for termination conditions. Similarly, it is useful to be able to compare visually the effect of various terminators.

One simple qualitative way to do this is to overlay the layouts and provide the user with the ability to perform relative rotations to align them. Individual nodes may be distinguished (e.g. by colour) to provide a reference. This is extremely easy in VRML.

While one could pin corresponding nodes in multiple layouts and force them to remain co-located, it is difficult to select in advance the most suitable nodes for this purpose.

Figure 2 shows a superior approach for visual comparison. The algorithm has been run twice with identical conditions apart from the initial random node placement. The resulting layouts have been aligned so that their centres of mass, rather than any individual node, coincide and for relative rotations to be about their common centre of mass. Figure 2 shows two layouts of a hypercube in reasonable alignment and two layouts of a web trail before user-controlled alignment begins. This approach is very successful for providing qualitative assessment of the layout quality achieved by algorithm and terminator choices and is also useful for detecting isomers.

4 ANGLE

ANGLE is a Java application developed to provide a platform for our experiments. Our requirements included the ability to repeat layout experiments in order to gather statistical data for quality evaluation, select a layout algorithm or force set, choose an appropriate terminator, control the appearance of the resulting world, observe and replay the algorithm steps and add new algorithms and terminators with minimal effort. The architecture of ANGLE is shown in Figure 4.

```

<graphset>
  <graph>
    <title>Tetrahedron</title>
    <nodes>
      <node><name>nA</name><coord>0 0 0</coord></node> <node><name>nB</name></node>
      <node><name>nC</name></node> <node><name>nD</name></node>
    </nodes>
    <edges>
      <edge><from>nA</from><to>nB</to></edge> <edge><from>nA</from><to>nC</to></edge>
      <edge><from>nA</from><to>nD</to></edge> <edge><from>nB</from><to>nC</to></edge>
      <edge><from>nB</from><to>nD</to></edge> <edge><from>nC</from><to>nD</to></edge>
    </edges>
  </graph>
</graphset>

```

Figure 3: NGML description of tetrahedron graph

Graphs are described in NGML an XML-based representation including nodes and edges together with some other details such as text descriptions. Figure 3 shows the NGML description of a tetrahedron. The initial position of node `nA` is specified explicitly by a `coord` element, a useful feature when we need to compare the behaviour of different algorithms on identical initial configurations or to force a planar layout.

One output is a VRML world containing the layout resulting from the choices made. The appearance of the nodes and edges, including such properties as the shapes and colours of nodes and the cross-section and thickness of edges, is determined by the PROTO nodes in the VRML template selected.

The available algorithms and terminators are simply concrete implementations of corresponding abstract classes. This approach allows ready extensibility and the menus in the GUI are constructed dynamically from all available concrete classes.

Figure 5 shows ANGLE in action on a graph made by forming a 4×25 rectangular grid into a Moebius strip. The interface consists of two main windows, one containing the projection of the current layout onto the xy plane and the other containing a control panel. Each algorithm or terminator specifies any parameters required and corresponding slider controls are then automatically constructed in the ANGLE control panel.

The control panel visible in Figure 5 shows the controls for the F3D force set (force constants k^a and k^r and natural spring length l_0) and the user is in the act of selecting a terminator. This will result in the addition of further controls specific to the selected terminator. Controls for zooming the layout window, setting the upper limit for iteration number and replaying the layout

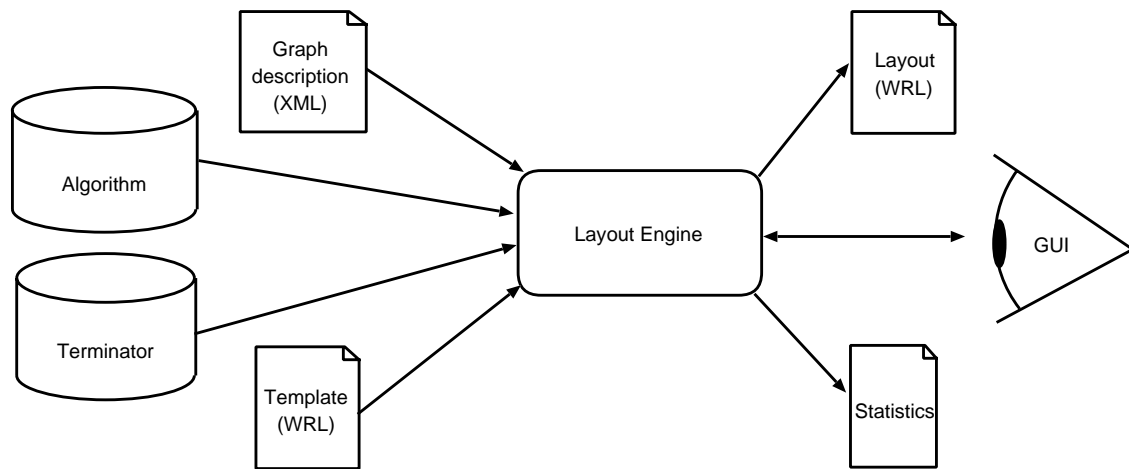


Figure 4: Angle architecture

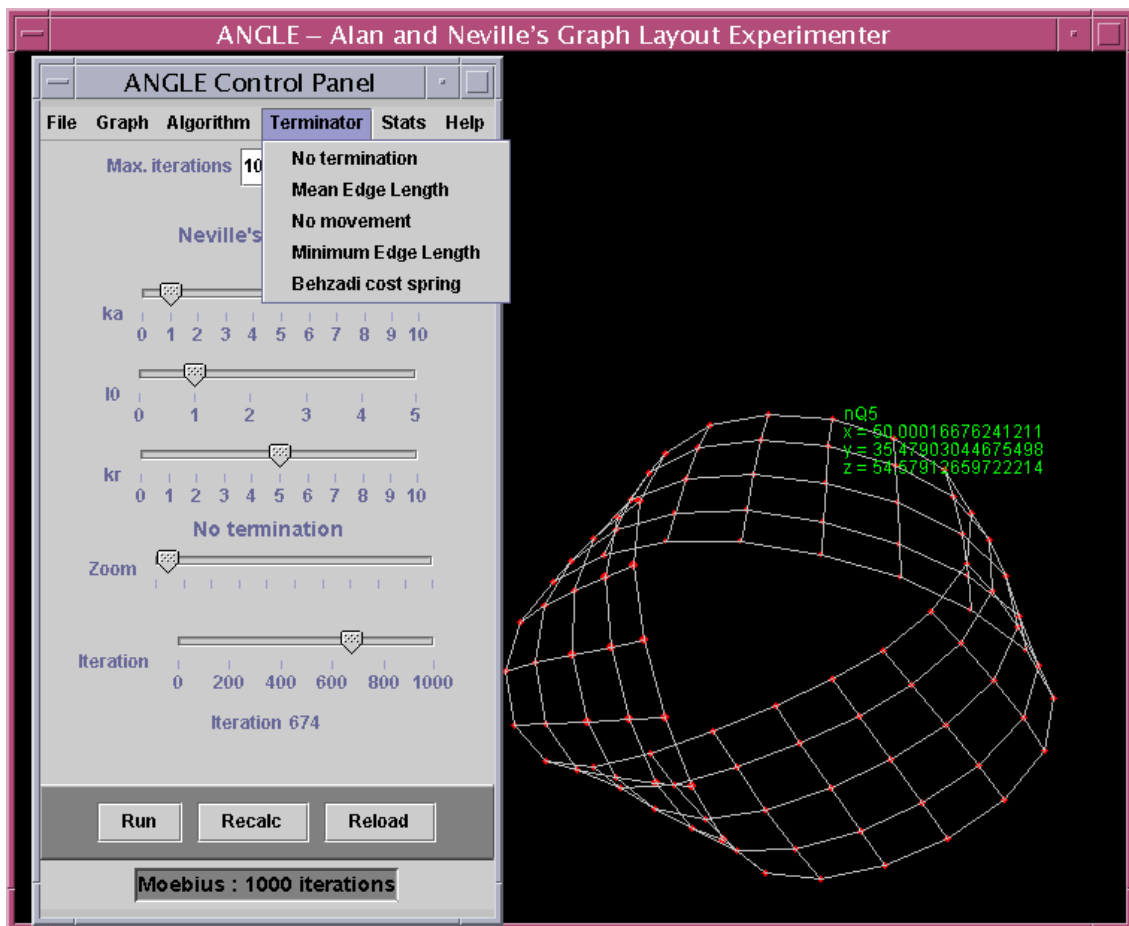


Figure 5: Angle application interface

process are always present. Selecting individual nodes in the layout window causes details of their precise locations to be displayed.

The *iteration* slider is used to replay a run of a algorithm forwards or backwards. This provides valuable information about trends in convergence and the larger scale behaviour of individual algorithms. For example, Fruchterman & Reingold’s force set often leads to oscillatory behaviour and the F3D set [8] may exhibit slow expansion in the latter stages.

5 The big-bang model

Our initial experiments led to some insight into the way the algorithms performed for different kinds of graph. For many graphs, the progress of the layout algorithm may be characterised by an initial *primary* phase, during which the major features of the final layout are established, followed by a *secondary* phase, during which relatively minor corrections are made. During the primary phase, the relative magnitudes of the attractive and repulsive forces are broadly comparable and the graph settles into a layout recognisable as that of the final layout. During the secondary phase, nodes wiggle a little, longer linear structures straighten gradually, surfaces flatten and free child nodes form radial structures.

Ideally, termination criteria will focus on ensuring that all of the primary phase is completed and that as little as possible of the secondary phase is carried out. The job, again ideally, of the layout algorithm is to ensure that the transition between phases is as abrupt as possible.

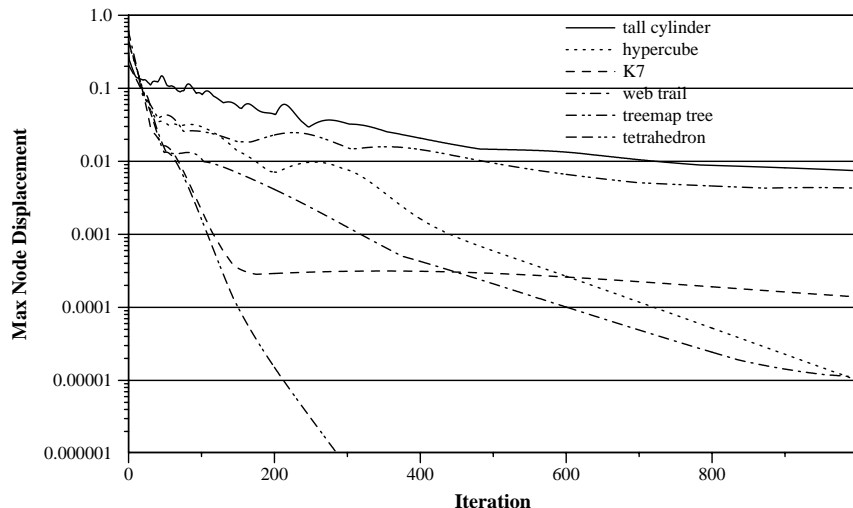


Figure 6: $\max(|\Delta \mathbf{n}_i|) < \epsilon$ for single F3D runs of 1000 iterations

Figure 6 shows some typical data from individual experiments with ANGLE. The tetrahedron data has been truncated—the maximum displacement value drops steeply to 10^{-16} after 1000 iterations. Force methods typically perform well on small highly connected structures.

The K7 data illustrates clearly a sharp distinction between phases. After around 200 iterations, progress is very slow. The hypercube and web trail show an initial steep drop but after around 100 iterations the rate of decrease declines. Performance is still good—the vertical axis is logarithmic.

The tall cylinder and treemap tree are clearly the worst performing. The initial drop is relatively shallow and progress is slow while gradual straightening occurs. During the secondary phase for such problematic graphs, the *local* structure (pattern of squares and pentagonal cross-section of cylinder; basic node structure on tree) is well established (see Figures 7(e), 7(i) and 7(g)) and the changes are more global in nature being dominated by repulsion of *groups* of nodes.

While the weak inverse-square repulsive forces have relatively short range, the Hooke’s law attractive forces have unbounded range. One method of attack is to introduce longer range $\propto r^{-1}$

repulsive forces [8] or to weaken the attractive forces [6]. However, our experience suggests that tampering with the “natural” forces can lead to undesirable behaviour such as oscillation requiring in turn artificial remedies such as cooling.

Our experience, guided by our ability to replay layouts in progress, led to an alternative approach. If the secondary phase is characterised, in problem cases, by slowly converging weak repulsions between distant nodes then we would do better to avoid such a situation arising. Rather than tinker with the functional form of the forces, we would prefer to concentrate on separating phases clearly and increasing the chances that the secondary phase is dominated by attractive rather than repulsive forces. A crude analogy might be to suggest that we boil vigorously before beginning the layout algorithm rather than cool gently during it.

One way to achieve this would be to modify the functional form of the forces in Equation 1 so that they depend on the current iteration number as well as the other parameters.

For example, the choice

$$\mathbf{F}_{ij}^r(n) = \begin{cases} k^r \hat{\mathbf{r}}_{ij} & 0 \leq n \leq n_{BB} \\ \frac{k^r \hat{\mathbf{r}}_{ij}}{|\mathbf{r}_j - \mathbf{r}_i|^2} & n > n_{BB} \end{cases} \quad (3)$$

causes the repulsive forces to be of constant magnitude for the first n_{BB} iterations and to follow the conventional inverse-square behaviour thereafter.

By varying the form of the forces we introduce an initial “big-bang” phase during which we boost the effect of the repulsive forces. After n_{BB} iterations the subsequent phases are initially dominated by the long-range attractive forces. Stretching the physical analogy to its limit we can argue that this phase corresponds to the initial big-bang where the geometry of space-time was still forming and where strange forces unlike those found today were at play.

Graph (N, E)	F3D				Big Bang				$\Delta\%$
	μ	σ	Min	Max	μ	σ	Min	Max	
tetrahedron (4, 6)	150.65	23.67	124	203	54.1	13.46	30	90	64
cube (8, 12)	261.45	36.14	212	350	93.25	9.87	85	118	64
hypercube (16, 32)	280.00	43.83	203	364	200.95	25.07	119	222	28
dodecahedron (20, 30)	503.8	87.46	344	691	226.55	10.85	217	254	55
tree (20, 19)	1104.55	212.26	715	1579	933.65	251.26	671	1489	15
grid 5×20 (100, 175)	2581.55	901.97	1669	4979	2951.3	12.72	2919	2965	-14
squat cyl. (100, 105)	1889.45	407.74	1201	2545	1345.2	16.63	1330	1389	29
tall cyl. (100, 120)	1973.55	338.71	1399	2529	2898.5	24.67	2834	2905	-47
web trail (15, 55)	250.10	52.53	183	370	203.75	29.95	158	255	19
inheritance (35, 34)	1680.4	202.79	1280	2223	1329.85	301.02	945	1932	-21
treemaptree (23, 22)	991.9	56.72	908	1145	948.85	171.01	705	1243	-4
K7 (7, 21)	103.10	16.6	81	147	18.75	2.55	16	26	82

Table 1: Comparison over 20 runs of F3D and BB with terminator $\max(|\Delta \mathbf{n}_i|) \leq 0.005$

6 Results

Tables 1 and 2 show results (mean, standard deviation, minimum and maximum number of iterations) obtained from 20 independent runs of each method with all parameters fixed apart from the initial random node placement within a $200 \times 200 \times 200$ cubic volume. The F3D algorithm uses “natural” forces (inverse-square repulsion; Hooke’s law attraction with $l_0 = 1$). The big-bang (BB) version has the repulsive forces of Equation 3 with $n_{BB} = 8N$ and Hooke’s law attraction. No cooling is applied in either case. The terminator ($\max(|\Delta \mathbf{n}_i|) < \epsilon$) is used with ϵ set to 0.005 spatial units (nominally 5mm) in Table 1 and to 0.001 in Table 2. An upper limit of 10000 iterations was imposed in all cases.

Graph (N, E)	F3D				Big Bang				$\Delta\%$
	μ	σ	Min	Max	μ	σ	Min	Max	
tetrahedron (4, 6)	241.45	51.70	172	362	70.40	11.53	58	97	71
cube (8, 12)	395.50	50.54	314	478	106.00	10.05	92	126	73
hypercube (16, 32)	462.55	137.85	323	872	242.25	12.04	215	267	48
dodecahedron (20, 30)	783.05	89.27	612	1022	251.65	12.24	235	274	68
tree (20, 19)	2600.75	608.61	1875	3525	1807.05	564.20	1322	3517	31
grid 5×20 (100, 175)	10000	0	10000	10000	3374.40	12.15	3348	3391	66
squat cyl. (100, 105)	3639.20	882.61	2293	5000	1484.15	12.15	3348	3391	59
tall cyl. (100, 120)	10000	0	10000	10000	3309.60	15.46	3279	3333	67
web trail (15, 55)	859.65	351.47	379	1825	240.80	41.34	188	332	72
inheritance (35, 34)	3774.20	954.80	2205	5000	3673.40	808.11	2153	5000	3
treemap tree (23, 22)	2643.40	588.32	1713	3713	2095.35	357.56	1410	2728	21
K7 (7, 21)	152.10	28.75	102	201	25.05	4.22	18	35	83

Table 2: Comparison over 20 runs of F3D and BB with terminator $\max(|\Delta \mathbf{n}_i|) \leq 0.001$

Layouts corresponding to individual runs summarised in Table 1 are shown in Figure 7.

Some trends are evident. For a given algorithm more iterations are required for $\epsilon = 0.001$ than for $\epsilon = 0.005$ and the relative improvement of the BB approach is greater. Where the BB results appear worse than for F3D the resulting layouts are still visibly superior.

Our BB approach appears to deliver consistently substantial improvements over the basic F3D method. We can calibrate it on regular polyhedra where the “right” layout is clear. Performance is also encouraging for graphs arising in experimental work such as the web trail [9]. In general, we find that, like all force methods, BB performs significantly better than F3D on “closed” graphs with few hinge points.

Graphs such as the 4×25 rectangular grid, an example from Behzadi [1, 2], and the two cylinders formed by joining its edges are particularly challenging for force methods as discussed in Section 5.

However, this is an ideal application for the BB approach. The data from Table 1 suggests that F3D outperforms BB on the grid and tall cylinder. However, as can be seen from Figures 7(e)–7(j), the resulting layouts for BB are superior. Although both have met the termination conditions, this simply confirms the very slow progress being made in the F3D case. For the smaller ϵ case the BB method performs well while the F3D algorithm has not terminated after 10000 iterations.

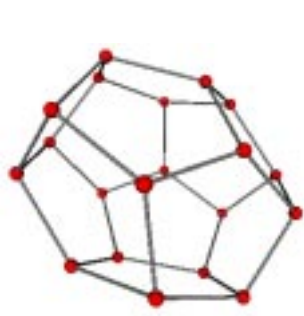
Three tree structures are included: one (tree) artificial, one (treemap tree) used as an example elsewhere [10] and one (C++ inheritance) arising in a research project on software reuse.

Figures 7(k) and 7(l) show treemap tree layouts produced during the runs summarised in Table 1. The BB method, Figure 7(l), leads to straighter spines for tree structures. This is because the effect of repulsive forces between clusters of child nodes is heightened during the BB phase. In contrast, the F3D method, Figure 7(k), generates layouts which straighten only very gradually. The radial features and straight spines discussed in Section 5 are evident.

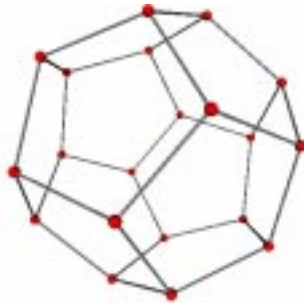
The effect of the big bang approach is to allow the fundamental spatial relationships between nodes to influence more strongly the initial stages of the layout, increasing the likelihood that the overall configuration will be roughly “right” before the secondary phase begins. The next phase is then characterised by a snap to the final size and minor rearrangement of nodes relative to each other. This process is clearly visible when the layout is replayed in ANGLE.

7 Conclusions and future work

Our ANGLE application provides a platform for experimenting with layout algorithms and is particularly valuable in our work with 3D layout algorithms and the generation of layouts as virtual worlds.



(a) dodecahedron F3D



(b) dodecahedron BB



(c) webtrail F3D



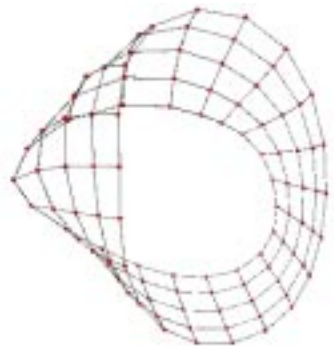
(d) webtrail BB



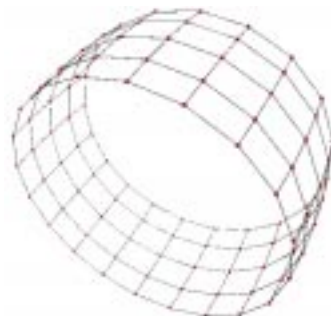
(e) grid F3D



(f) grid BB



(g) squat cylinder F3D



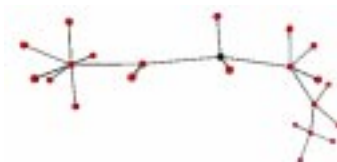
(h) squat cylinder BB



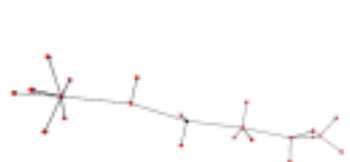
(i) tall cylinder F3D



(j) tall cylinder BB



(k) treemap tree F3D



(l) treemap tree BB

Figure 7: Snapshots of 3D virtual worlds corresponding to the data of Table 1

The big-bang approach was proposed as a reliable and efficient way to get acceptable layouts for a wide range of graphs. At the macro level, we can perform both static and animated comparisons of the layouts produced by BB against those from other techniques and observe the dependence of this comparison on the number of iterations. At the micro level, we can observe properties such as convergence and the effect of terminators.

Our results are encouraging. We intend to incorporate BB techniques into our current visualisation projects. The visualisation of layout algorithms is itself a topic of considerable interest to us. We are also investigating the opportunities for extending the application of our methods to diagramming techniques such as UML.

References

- [1] Lila Behzadi. An improved spring-based graph embedding algorithm and layoutshow: A java environment for graph drawing. MSc thesis, York University, Ontario, Canada, 1999.
- [2] Lila Behzadi. LayoutShow: A signed applet/application for graph drawing and experimentation. In Jan Kratochvíl, editor, *Graph Drawing: Proc 7th International Symposium GD'99*, volume 1731 of *Lecture Notes in Computer Science*, pages 242–249, Štířín Castle, Czech Republic, September 1999. Springer-Verlag.
- [3] R. Carey and G. Bell. *The Annotated VRML 2.0 Reference manual*. Addison-Wesley, 1997.
- [4] N.I. Churcher, L.M. Keown, and W. Irwin. Virtual worlds for software visualisation. In A. Quigley, editor, *SoftVis99 Software Visualisation Workshop*, pages 9–16, University of Technology, Sydney, Australia, December 1999.
- [5] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
- [6] Peter Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.
- [7] Peter Eades, Mao Lin Huang, and Junhu Wang. Online animated graph drawing using a modified spring algorithm. Technical Report 97-05, Department of Computer Science and Software Engineering, University of Newcastle, Callaghan 2308, Australia, 1997.
- [8] T.M.J. Fruchterman and E.M. Reingold. Graph drawing by force-directed placement. *Software—Practice and Experience*, 21:1129–1164, 1991.
- [9] D. Hartley, N. Churcher, and G. Albertson. Virtual worlds for web site visualisation. In *Proc APSEC2000, 7th Asia Pacific Software Engineering Conference*, pages 448–455, Singapore, December 2000. IEEE Press.
- [10] Brian Johnson and Ben Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In G.M. Nielson and L. Rosenblum, editors, *proc. Visualization '91*, pages 284–291, Los Alamitos, CA, October 1991. IEEE Computer Society Press.
- [11] M. Kaufmann and D. Wagner, editors. *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
- [12] C Ware and G Franck. Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Transaction on Graphics*, 15(2):121–139, 1996.