Graph Drawing in Lightweight Software: Conception and Implementation

Vitaly Zabiniako
Riga Technical University, Institute of Applied Computer Systems,
1/3 Meza street, Latvia,
LV-1048, Riga.
Vitalijs.Zabinako@rtu.lv

Pavel Rusakov
Riga Technical University, Institute of Applied Computer Systems,
1/3 Meza street, Latvia,
LV-1048, Riga.
Pavels.Rusakovs@cs.rtu.lv

ABSTRACT

This work describes basic ideas of lightweight graph visualization system developed in Riga Technical University. Comparison of according existing freeware and shareware solutions is being made. Overall proposed software architecture at high abstraction level is presented along with details of implementation of its mechanisms. This work includes aspects of optimization of force-based graph layout algorithm; description of useful visualization techniques (such as projective shadows, visual data clustering that might be useful in design and analysis routines, etc.). Described ideas were implemented and verified by visualization of large graphs in original lightweight software 3DIIVE. Conclusions about achieved results are also presented.

Keywords

Graph, visualization, architecture, algorithm, clustering.

1. INTRODUCTION

The concept of information visualization plays important role in modern IT industry, as it allows representing data according to the current needs of end-user and information processing tasks. There is a demand for these tools in such domains as data mining and analysis, education process, etc.

Multiple visualization solutions already exist, each with its own functionality and implementation specifics. Some of these tend to handle wide range of input data types and visualization tasks (which comes at the cost of complexity and usually – bulky architecture), while others are more of ad-hoc type solutions for specific purposes (and, as a result, non-usable outside originally intended visualization domain). The authors of this paper argue that there is a need for more agile solutions that should be based on achievements of modern computer graphics and object-oriented approach. These will provide appropriate aid for users in science, industry and business. Proposed architecture must ensure a set of primary features (i.e. ability to store and represent topology and associated metadata of a graph using appropriate description formats, layout algorithms and visualization techniques for better comprehension).

2. OVERVIEW OF EXISTING GRAPH DRAWING SOLUTIONS

Nowadays one can find wide range of tools that provide aid in graph drawing. Considering that it is impossible to summarize features of all these tools and versions, authors decided to analyze few distinctive representatives from both freeware and commercial products (summary of main aspects of such tools are presented in Table 1). Additionally, authors choose a set of criteria that is

Criteria / Solution	Data format	Space	Graphics library	Platform	Additional tools
Graphviz	DOT (plain text-based)	2D	Native	Windows / Linux / Mac OS	–
Wilmascope 3D	Native XML-based, GML	3D	Java 3D	Windows	+
aiSee	GDL	2D	Native	Windows / Linux / Mac OS	–
Tulip	Native, GML, DOT	3D	Native	Windows / Linux	+
yFiles	Native, GraphML, GML	2D	Native	Windows / Linux / Mac OS	+
Walrus	Native	3D	Java3D	Windows / Linux / Mac OS	–
Tom Sawyer	Native	2D	Native	Windows / Linux / Mac OS	+
VGJ	GML	2D	Native	Windows / Linux	–
3DIIVE	Native XML-based	3D	OpenGL	Windows	+

Table 1. Comparison of graph drawing solutions

based on aforementioned primary features, support of additional analysis tools and implementation capabilities and allows comparing these solutions in order to get general vision of functionality.

3. OVERALL SOFTWARE ARCHITECTURE AND INDIVIDUAL FRAMEWORK COMPONENTS

Last row of Table 1 contains information about original lightweight graph visualization software system that is being developed in Riga Technical University for academic purposes as a part of doctoral thesis research. This system will be referred as "3DIIVE" (Three–Dimensional Interactive Information Visualization Environment) further in this text. This system was initially intended as utility program for demonstrating basic concepts in graph drawing area (focusing on drawing in three dimensions). As it can be perceived from the table, 3DIIVE is an agile solution that might be used both for general visualization of information encoded in form of graphs and more specific data-oriented applications in MS Windows platform. High abstraction level of proposed software framework is presented in Fig.1.

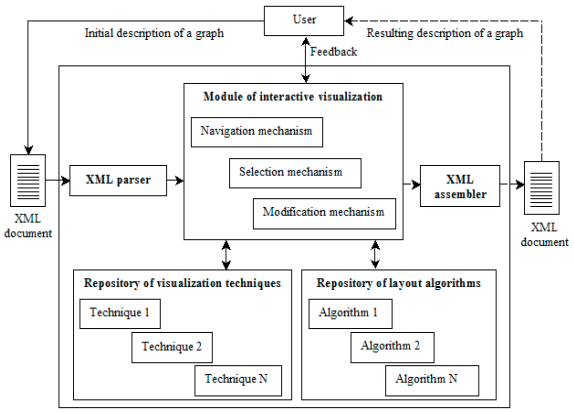

Fig.1. Architecture of 3DIIVE system

Our approach is based on the assumption that there must be clear separation in functionality of layout algorithms and visualization techniques – in this case these will be able to perform independently, making implementation of other algorithms / techniques much easier. Another assumption is that the structure of proposed framework must conform to the object-oriented approach, because graph itself can be conveniently interpreted as a set of topological and other associated properties that can be altered by appropriate methods.

Considering aforesaid, the main parts of this system are as follows: *XML parser* (interpretation of the input XML document with a description of the graph, and extraction of necessary information about topology of the graph and its elements), *module of interactive visualization* (the main part of the system that performs visualization by triggering necessary layout algorithms and visual techniques from repositories and relies on navigation, selection and modification mechanisms), *repository of layout algorithms* (a set of available layout algorithms for visualization of graphs), *repository of visualization techniques* (a set of techniques for improvement of comprehension) and *XML assembler* (acts similar to XML parser with only difference – opposite information processing flow).

The graph within a system is presented as a separate object with multiple attributes and methods. Information about topology of the graph from the parser module is converted into the adjacency matrix. The rest of the data is loaded into multiple arrays with references to associated elements.

4. IMPLEMENTATION OF LAYOUT ALGORITHMS AND VISUALIZATION TECHNIQUES

4.1 Layout algorithms

Current implementation relies on well-known force-based layout algorithm proposed by Peter Eades [Ead84]. The repository holds both its original and custom versions with optimizations done by authors. It is founded on combination of force-based and orthogonal layout properties and includes additional improvements.

For example, during the search of equilibrium state, in case if a value of average kinetic system energy achieves local minimum, it takes an additional time to retrieve from this, that's why authors make a step with random offset from current position, bypassing "slow" iterative recovering upon detection of local minimum. Another aspect is that the equilibrium state is being formed with unpredictable offset in space. In our system the model of graph is placed near the origin of a coordinate system.

Authors evaluated performance of both original ("FB") and modified ("MOD") algorithms with a set of time measurements from the first iteration and to the moment, when an equilibrium state was reached. Probability distribution is shown in Fig.2.

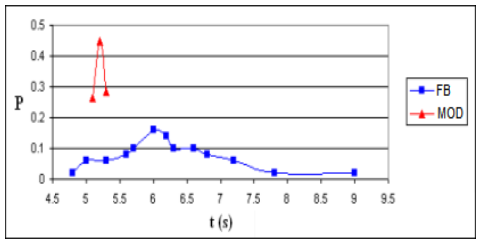

Fig.2. Probability distribution graph

Statistical processing allows concluding that the modified algorithm performs about to 15% faster while showing more stable distribution of time required for the execution. For detailed description of improved algorithm and according experiments refer to [Zab08].

4.2 Visualization techniques

Repository of visualization techniques holds a set of tools for improvement of information comprehension. It includes both common useful techniques, such as *transparency* (in case if certain part of the graph is chosen for further analysis, transparency helps to abstract from the other data by visually "weakening" it while still allowing to perceive the whole graph structure), *magnification* (interactive scaling up visuals that is implemented via image post-processing and allows to see more details by increasing resolution of these), etc. For more detailed description refer to [Zab09].

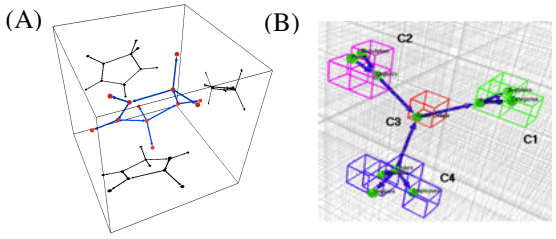

Fig.3. "Projective shadows" and "Visual clustering"

In 3DIIVE there are few custom visual techniques implemented by authors. One of such techniques is "*Projective shadows*" (Fig.3, part A). The idea is to draw the three-dimensional data model in iterative steps. The first step captures the model from the current position of the virtual camera like in previously mentioned techniques. During next steps camera is placed so that it faces model orthogonally – directly from the top, front, left etc. Each rendering result is placed into separate texture. When all steps are complete, textures are placed on corresponding faces of the rectangular parallelepiped (or cube). The parallelepiped is drawn in the scene so that three-dimensional data model is situated at its centre. In this case each texture represents a projection or essentially a "shadow" of original data structure.

As the result, user perceives not only the graph itself, but he can also evaluate and choose one two-dimensional instance which suits for outputting it in a plane surface (for example – for printing the graph on a paper). Two-dimensional instances are updated each frame and modifications that user performs with the spatial graph are reflected in all projections in real-time which makes this technique particularly useful.

In order to support this technique, graphical framework must provide access for rendering to texture which can be applied to the scene later. OpenGL framework (this API has been chosen due to its simple yet effective state machine model [Zab06]) allows to implement projective shadows with the code as follows:

```
/*1*/ ... //set matrix for camera
/*2*/ ... //draw object
/*3*/ for (x=1; x<=DimNum; x++)
/*4*/ {
/*5*/ ... //set matrix for dimension
/*6*/ ... //draw object
/*7*/ glCopyTexImage2D(GL_TEXTURE_2D,
          0,GL_RGBA,0,0,512,512,0);
/*8*/ }
/*9*/ //draw cube with textured faces
```

Projective shadowing allows to get more detailed comprehension about data structure – when perceiving spatial model from particular point of view it is sometimes hard enough to judge how complex the graph topology in individual dimensions is. Multiple projections allow to get rid of this problem. This concept is similar to orthographic projection views in CAD (Computer-Aided Design) systems. The difference is that in this case shadows allow to explore topological relationships of multiple elements rather than purely geometrical properties of single object. It is also suitable for tasks where the result of visualization must be presented or printed out as planar image.

Another custom implemented technique is in relation with graph data *clustering* task (Fig.3, part B). In general, clustering is required in case if data units must be evaluated in terms of its similarity or semantic closeness [Wri04]. Two common examples of such analysis are optimization of storing data in memory (in the field of database technology classic implementation of "Many-to-Many" relationship between two entities via auxiliary table results in data storage in different memory regions, so unwanted additional fetching of memory pages would be required) and deriving hidden patterns in large loosely structured data sets (detecting of dense semantic relationships among entities of knowledge domain may influence management strategy and even trigger development of new business rules).

Data clustering opportunity in 3DIIVE is based on secondary effect of force-based approach: the final layout tends to group densely interconnected nodes close to each other, while separating loosely connected groups in different space regions. A space partitioning mechanism is required to get the desired results (separate sets containing unique data

elements). The implemented model of space partitioning is based on octree that produces recursive division of cube model as shown in Fig.4.

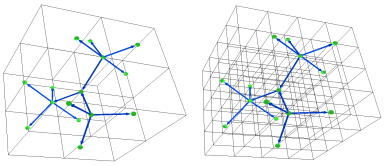

Fig.4. Recursive space partitioning model

Finally, the last implemented mechanism that is needed for accomplishment of clustering is regions merging – the natural way to bring multiple close nodes into corresponding cluster. Definition of a cluster with a set of neighbor regions is recursively transitive by its nature – region *R*, its neighbors *{R'}*, neighbors of neighbors *{R''}*, …, etc. belong to the same class. Authors propose the pseudo-code for the algorithm for assigning unique cluster identifications for an ordered set of regions:

```
proc assign_cluster_id
  for each region r process_region(r);
end

proc process_region(r)
  if r = Ø or r has id then return;
  if r has neighbors with id then
        assign same id to r;
  else
        assign new id to r;
  end
  for each neighbor n process_region(n);
end
```

The first procedure *assign_cluster_id* initiates sequential processing of regions by calling subroutine *process_region*. When non-empty and un-marked region has been found, it becomes either a core of new cluster or a part of already existing one – depending on the state of surrounding regions. Then all its neighbors are recursively revisited as in Fig.5 (sequence of regions is marked with red numbers and resulting clusters are marked with blue).

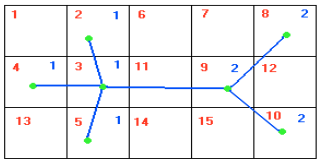

Fig.5. Example of partitioning sequence

5. USAGE EXAMPLES

Screenshots of different 3DIIVE functioning modes while visualizing complex network that consists of few hundred nodes are presented in Fig.6, (part A – selection step, B – usage of magnification, C – usage of transparency, D – clustering step).

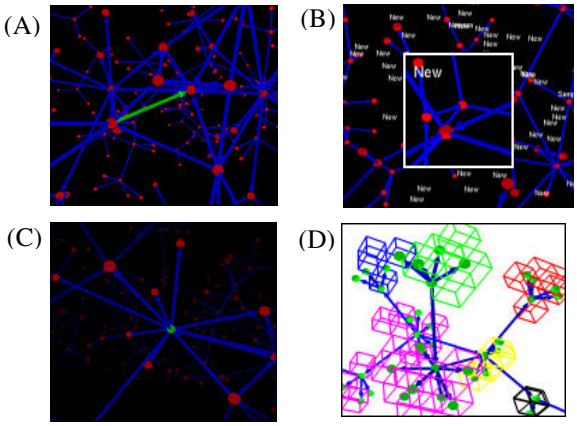

Fig.6. Screenshots of 3DIIVE workflow

6. CONCLUSION

Our implementation – 3DIIVE software system, allows visualization, analysis, editing of general graphs and provides a set of useful techniques for better information comprehension.

Current version holds both well-known existing graph drawing solutions and those proposed by the authors. The functional content of each module is gradually expanded during author's researches in the domain of graph visualization.

Considering its clustering capabilities this system can be used not only as a system for visualization and analysis but also as a tool for optimization, e.g. for refining relational model of database structure.

7. REFERENCES

[Ead84] P. Eades. A heuristic for graph drawing, Congresses Numerantium, vol. 42, pp. 149-160 (1984).

[Wri04] M. Kaufmann, D. Wagner "Drawing Graphs: Methods and Models", Springer, 312 p. (2001).

[Zab06] V. Zabiniako, P. Rusakov. Comparative Analysis of Visualization Aspects in Technologies Direct3D and OpenGL. Scientific Proceedings of Riga Technical University, Computer Science, series 5, vol. 26, pp 209-221 (2006).

[Zab08] V. Zabiniako, P. Rusakov. Development and Implementation of Partial Hybrid Algorithm for Graphs Visualization, Scientific Proceedings of Riga Technical University, Computer Science, ser. 5, vol. 34, pp. 192 – 203 (2008).

[Zab09] V. Zabiniako, P. Rusakov "Supporting Visual Techniques for Graphs Data Analysis in Three-Dimensional Space". The 50th Scientific Conference of Riga Technical University, Computer Science, Applied Computer Systems, October, Riga, Latvia (2009).