DDDiver:
3D Interactive Visualization of Entity Relationships

Marc Coomans and Harry Timmermans

Eindhoven University of Technology, Faculty of Architecture, Building and Planning,
Mail station 20, P.O. box 513, 5600 MB Eindhoven, Netherlands
{M.K.D.Coomans, H.J.P.Timmermans}@tue.nl

Abstract. In this paper we present DDDiver, a tool for the interactive visualization and editing of Object-Oriented databases. It was developed to visualize and manipulate large loosely-structured data sets with multiple relation types. This makes the tool especially useful in application areas that involve product data models, design information systems, and semantic networks. DDDiver can visualize such relational data sets in a 3d graph. The layout mechanism used for the graph is not based on a deterministic mathematical algorithm, but on the distinction between a number of relation kinds, and on user interaction. The intuitiveness and quickness of the visualization tool was further improved by adding animated visual feedback effects.

1 Introduction

Information visualization is one of the relatively new areas of research and development in computer science. The visualization of large, abstract data structures is often regarded as one of the crucial tasks in bringing computers closer to the general public [1]. Unfortunately, many commercial software applications that deal with complex relational data sets do not offer the appropriate visualizations, with which a user can easily observe the true structure of the data at hand. These poor visualizations are the result of both a lack of graph drawing knowledge in the field, and of the lack of graph drawing support in software development tools.

Some specialized systems have come to the fore in the last years which can visualize large complex data networks very well. Some examples of such systems are the H3Viewer[2], NicheWorks[3], and the LaTour system[4]. These are very useful tools for the visualization of web site structures, organizational diagrams, etc. Unfortunately they are much less useful for the visualization of product data models and semantic networks, since they focus on data sets in which all relations are of a single kind.

In the domains of product data modeling and knowledge modeling, relational data sets are constructed with several kinds of relations. These data sets usually also lack any rigid internal structure (e.g. no tree structure), and they can be subject to cyclic relationships. The systems with which these complex databases are developed, typically offer only limited visualization facilities like tables and treeview visualizations (showing only a single relation type at a time). It is expected that the user interfaces of

these applications would considerably be improved if a more suitable visualization technique was implemented.

The DDDiver system is an experimental visualization system for the manipulation of Object-Oriented databases that are non-homogeneous in terms of both objects and relations. The occurrence of multiple relation types was the starting-point for its visualization method. In the next section, we first give a brief overview of the existing visualization techniques. In section 3, we discuss the visualization method offered by DDDiver, and consecutively in section 4, we present a practical application of DDDiver in a CAD system.

2 Existing Techniques

In software applications, relational data sets are conventionally displayed either in table format or in 2D graphical layouts. Tables can display lists of object-relation-object sets in a very efficient way and are very readable at the same time (figure 1a). Manipulation of data in relation tables is typically done by editing names and tags in the table fields; either manually or by selecting possible values from dropdown menus. The disadvantage of tables is that they do not very well support the discovery

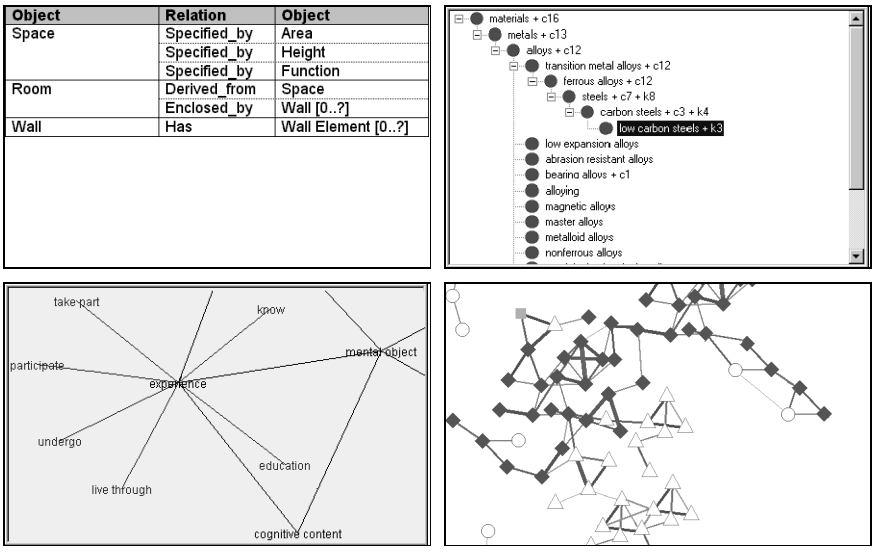

Fig. 1. Existing visualization methods for large relational data sets: a. relation table (top-left), b. treeview (top right), c. graph layouts with tags require lots of screen space (bottom left), d. large graphs are hard to visualize in an aesthetically appealing way (bottom right)

of implicit data structuring. Graphical layouts do provide a much better insight in the structure of the data sets. Object clustering and relation sequences can much easier be denoted.

The most used graphical layout in applications is the "treeview" (figure 1b). The treeview is well suited for hierarchical data sets. Its screen efficiency is close to that of tables. Non-hierarchical (cross-linked) data sets can best be visualized in graph lay-outs. One disadvantages of graphs layouts is that they are often less efficient in screen area usage, especially when nodes and edges have tags (figure 1c). Another is that cross-linked graphs tend to become messy and difficult to read when the number of objects and relations gets large (figure 1d).

The construction of aesthetically appealing graphs is addressed by a specific discipline of the mathematics and computer science community, known as "Graph Drawing" [5]. In spite of all the results achieved in this discipline, practical applications of large graph visualizations remain difficult. Most of the graph drawing techniques assume that the complete set of nodes and relations can reasonably be represented in a readable and understandable manner on the display medium [6]. Real-life applications deal with databases that are much too big to be displayed at once.

The visualization of real-life data sets requires interactive visualization techniques. At each moment only a small part of the total data set is displayed. The user is provided with navigation tools with which, in subsequent steps, details can be explored and unnecessary parts can be hidden.

When the classical graph drawing algorithms are used in practical applications, they are confronted with large data sets. In [4], the authors pointed out the 3 problems that often become problematic in these practical applications: (1) low speed due to the computational complexity, (2) lack of lay-out predictability for the user, (3) no navigation method supported.

Only a small number of visualization techniques have been presented that do not suffer from these problems. Graham [7] and Eades [6] presented two of these rare exceptions. Both have proposed a 2D drawing technique in which child nodes are located around parent nodes on respectively circles and circular wedges. In both cases, the spatial distribution of the child nodes around their parent depends on the context (number of the child nodes' children, …), and on the browsing history (which "uncle" and "nephew" nodes that have been looked at before).

Another restriction of the classical graph layout algorithms is that they are designed for networks that are homogeneous in both nodes and relations. Edge labeling is typically not taken into account either. Unfortunately, Graham’s and Eades’ layout mechanisms suffer from these restrictions as well.

3 The DDDiver Drawing Technique

The DDDiver system was developed as a visualization tool for product data models, design information databases, and semantic networks. In product data modeling, a multitude of product characteristics is collected in a single product database. The assembled information typically relates to multiple views on the product (technological, functional, …) and/or to multiple life-time stages of the product. The multitude of characteristics and information types is usually structured by the object orientation paradigm [8]. Besides the generalization/specialization relationship, product models

use the aggregation relations and the association relations. Semantic networks are widely used to represent structured knowledge. The nodes in a semantic network are concepts of a specific knowledge domain. The relations between the concepts are also formalized, typically making use of the following relation types: equivalence relations, hierarchic relations, scope notes, and associations.

In each of these application domains, object oriented databases are constructed that make use of a small number of relationship types with different semantics. This semantic difference requires these relations to be visualized in a clearly distinctive way. Further, nodes are meaningless without labeling. In product design modeling, edges also need labeling (on top of the visualisation of their relationship type).

One possible visualization solution would be to use a standard form of graph drawing, e.g. a spring-embedder model, and add labels on nodes and edges, and polish it up with color and shape to distinguish object and relation types. However, for the applications we have in mind, relation type differences are very important. Therefore, such a simple color or line-shape polishing is not sufficient. We want the user to be well aware of the semantic difference between browsing into, i.e., an object's specification list and browsing into that object's component list. Relation type differences are emphasized much better when they are reflected in the spatial layout, and not only in color and linetypes.

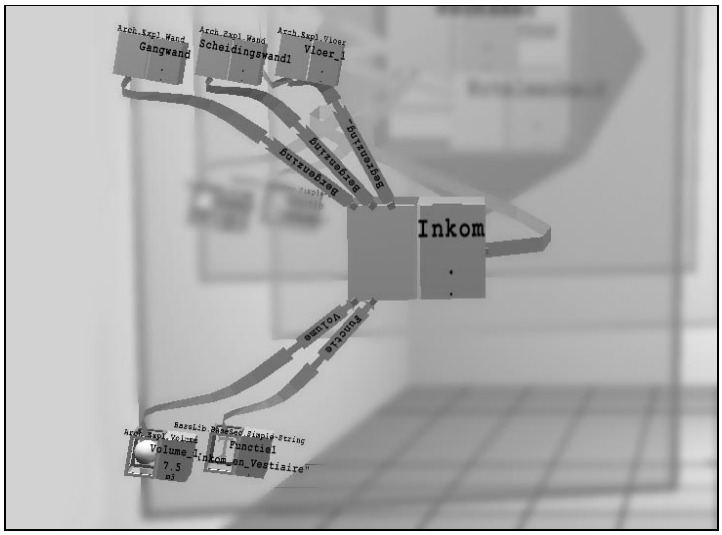

Fig. 2. The visualization of relation types in DDDiver, applied in a CAD system. The central object represents a room "Inkom". It has 2 specification-relations (directed to the bottom-left) that specify the volume and the function of the room. It also has 3 association-relations (directed to the top) with 3 other building components. Finally, it has a single aggregation-relation (horizontal link) with an object in the back (which is the building object).

In DDDiver, the relations of a node are grouped by type, and each group is given a characteristic direction. The recognizability of a relation's type is further improved by

distinct arrow styles and colors. The relations themselves are straight by default, but bent into S-like shapes when the connected nodes are not in line. When bent, the relation-end-arrows preserve the characteristic direction and the rest of the relation forms a fluidly connecting spline. See figure 2.

In DDDiver, both relations (edges) and objects (nodes) can have multiple tags. The visualization of a single object with all direct relations can therefore already consume a rather big screen area. Nevertheless, we wanted to be able to visualize a maximum of context for any node in focus. We developed a network browsing technique using multiple transparent layers. Lieberman demonstrated how multiple translucent layers can be used to provide both context and focus views on huge maps [9]. We adapted Lieberman's principle towards a browsing mechanism for graphs. Unlike Lieberman's application, DDDiver visualizes the transparent layers as 3D shells in 3D workspace. Shells popup in front of a 3D cubic box that represent the database. Data objects retreived from the database are always located on one of the shells.

Each shell can be divided into multiple fields. Each field can host a single data object and its directly related objects. (See figure 3.) When the user asks to see additional data of a related object, this object first moves to a new empty field of its own. This new field is by default located on an empty layer that is laid on top. On this new layer, the additional data objects are displayed around the central object that the user pointed out. The relations between the objects on the different layers stay intact through the spline deformation of the relation-bands, as explained above.

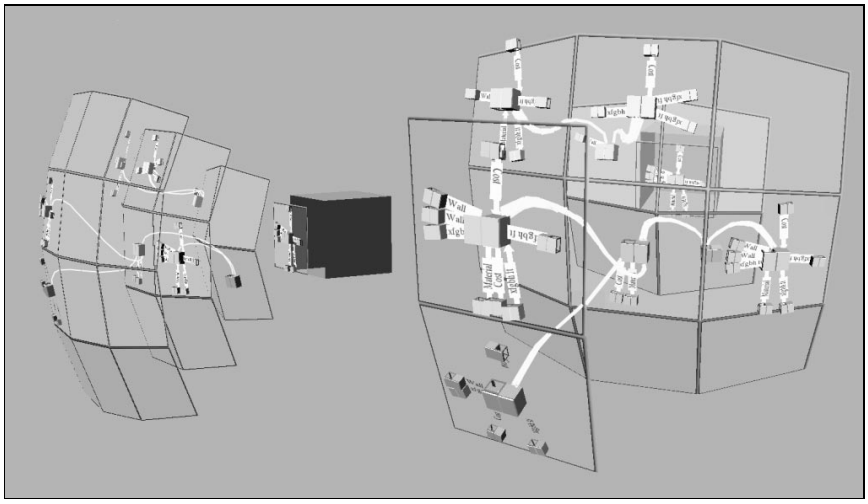

Fig. 3. DDDiver's transparant layers in front of the database box. In this example spherical layers were used with an increasing number of data fields

After multiple browsing steps, a pile of layers is formed. When a new layer is added, existing layers automatically move one step backwards. The bottom layers are automatically cleaned up and removed when the maximum number of layers is reached. We found that 5 to 6 layers is the maximum number of layers that is practically useful.

The top layer always shows the lastly retrieved data objects; layers underneath contain context data. When the user wants to return to data on a lower level, the top layers and the data on it can be removed again. The layers can also individually be moved side- and upward in a side by side position. In our approach, the order of the layers can not be changed.

DDDiver can be configured to create both planar and spherical layers (Figure 3 shows an example of spherical layers). The two layer shapes have corresponding manipulation styles: respectively sliding and rotating.

The number of field subdivisions can be chosen in function of the available screen area. The maximum number of usable fields is mainly restricted by the readability of the text labels on relations and objects. On a screen resolution of 1280x1024 pixels, we experienced that a subdivision of 3 by 2 fields is the practical maximum when the links are labeled. For unlabeled links, the numbers can be doubled. The use of super-scene anti-aliased rendering would also reduce the required screen area per field.

4 A CAD application

DDDiver is implemented as an application-independent tool-set for database visualization. However, the original specifications were formulated on the basis of the needs for a new Computer Aided Design (CAD) system for the building and construction industrie. This CAD system is characterized by an innovative design-information modeling technique that has been developed by van Leeuwen [8]. The modeling technique makes use of an Object Oriented modeling technique, with two extensions. Firstly, a user (designer) can define own object types. Secondly, a user can add extra relations at the instance level. In this way, object relationsips can be modeled that are not shared by the other objects of it's kind.

This new way of design information modeling offers a unique freedom for the building designer. He or she can control how abstract design data is structured and stored. In this way, the designer is given the power to model concepts like conformity, contrast, and scale on the formal data level. With this new information modeling technique, we expect that the designers will be better capable of handling the complexity of linking diverse kinds of information involved in a design process.

On the other hand, this technique also puts the responsibility for the content of the CAD database entirely in the hands of the designer. In order to be able to enjoy the design freedom fully and at the same time handle the responsibility over the design database, a computer tool was needed that showed the precise content of the database, and that was easy and quick to interact with. DDDiver was developed to meet these demands.

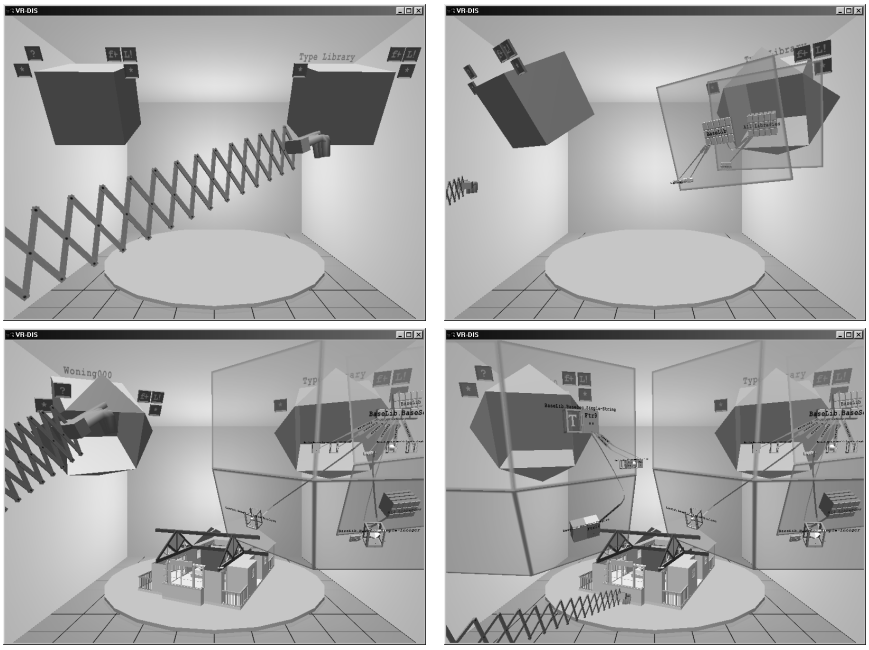

Fig. 4. In the CAD application, architects and construction engineers can directly interact with the desing database through DDDiver. Two DDDiver graphs were combined: one for the object type library (right side), a second for the object instances (left side). DDDiver's direct database visualization was further mixed with a mock-up style visualization of the design

The CAD system's database is made up of 2 parts: (1) the *type library* containing user customizable type definitions, and (2) a number of *desing models* that each describe a particular building design. A single type definition (in the type library) is small directed cyclic construct on itself, that can be composed out of 3 relation types (association, specification, component) and 9 basic node types. The design models contain the instantiations of these small constructs and form large directed, cyclic networks. These large design models can contain several thousands of nodes.

DDDiver was configured to display the type library and the instance database in 2 separate graphs that are positioned side-by-side; see figure 4, and color plate in the appendix. When looking for existing information, the browsing process works independent in both graphs. However, to instantiate objects, type nodes can be dragged from the type-graph to the instance-graph. The instantiating process is visualized as a duplication process after which the type node automatically returns to the type-graph, and the instance remains in the instance graph. To create a new user type, the inverse action can be performed: a set of instance nodes that are thought of as a prototype for the new type definition are dragged from the instance graph to the type graph. These and other user actions benefit from the built in 3D animation effects as these more clearly visualize the effects of a user's action.

In this application, the two DDDiver graphs were combined with a mock-up like visualization of the architectural design with a high level of verisimilitude. This pictorial mock-up representation complements the formal, descriptive representation in the DDDiver graphs. Notwithstanding that all data is already visualized in the graphs, and only a subset is visualized in the mock-up, such a mixed data representation provides a more complete view and thus a better understanding by the user [10]. An additional advantage is that the mock-up visualization now also can be used as an index for the formal data representation in the graphs: selecting an object in the mock-up (e.g. a wall) also pops-up the corresponding data objects in the graph. This can shorten a user's information retrieval time considerably.

5 Implementation

DDDiver is implemented as a function library for the VR software package World Up from Sense8 (a division of Unigraphics). It runs on the Windows™ platform (NT, 2000 and 98). The use of the DDDiver library is currently restricted to that software package.

The described CAD application was also implemented in World Up. The CAD database was implemented with PSE Pro from Object Design (a division of eXcelon). The database is accessed from World Up through Active-X functionality.

DDDiver requires a PC with 3D accelerated graphics board with OGL support. It runs smoothly on a recent PC with a Geforce™ graphics board.

6 Conclusions

We presented DDDiver, a tool for the visualization of relational databases that are used for product data modeling, design information databases, and semantic networks. Unlike other graph visualization tools, DDDiver explicitly supports multiple relation types and object types, as well as labeling for both relations and objects.

An interactive data exploration mechanism was developed which makes use of multiple transparent layers to maintain data context. The whole is worked out as an integrated 3D interactive visualization.

Unlike other graph drawing approaches, the visualization in DDDiver not based on a computationally insensitive lay-out algorithm. As a result, the system's performance is only a linear function of the size of the displayed database part; it is independent of the total database size.

Within layers fields, one step object-to-object relationships are visualized purely on the basis of the underlying database structure, notably the relation type and relation index. As a result, the graphical layout is predictable and thus easy to interpret by the user. This predictability is partially lost after subsequent navigation steps, because the data-objects are then spread out over multiple layers. It can be argued that as a result, the user will not be able to build a mental map of the overall database structure. In the

VR-DIS application, we showed how a second data representation can be integrated to overcome this problem.

Acknowledgement

This project has been conducted in the context of the VR-DIS research of the Eindhoven University of Technology. More details can be found at the VR-DIS web site [11].

References

1. Card, S.K., Mackinlay, J.D., Scheiderman, B. (eds.): Readings in Information Visualization, Morgan Kaufmann Publishers (1999)
2. Munzner, T., "Drawing Large Graphs with H3Viewer and Site Manager", In: Proceedings of the 6th International Symposium on Graph Drawing GD'98, Montréal, August 1998, Whitesides S. (Ed.), p. 384-393, Springer-Verlag (1998)
3. Wills, G.J.: "Niche Works – Interactive Visualization of Very Large Graphs", In: Proceedings of the Symposium on Graph Drawing GD'97, Springer-Verlag (1997)
4. Herman I., Melancon G., de Ruiter M., Delest, M, "Latour – A Tree Visualization System", In: Proceedings of the 7th International Symposium on Graph Drawing GD'99, September 1999, Kratochvil, J. (ed.), p. 392-399, Springer, Berlin (1999)
5. Graph drawing, GD : yearly international symposium 1992 -1999; proceedings as of 1995 by Springer-Verlag, Berlin
6. Eades, P., F. Cohen and M.L. Huang (1997) Online Animated Graph Drawing for Web Navigation. G. Goos, J. Hartmanis and J. van Leeuwen (eds.) 1997. Graph drawing, Proceedings of the 5th International Symposium on Graph Drawing, GD'97, held in Rome, Italy, Sept. 18-20, 1997. Springer-Verlag, Berlin, pp. 330-335.
7. Graham, J.W. (1997) NicheWorks – Interactive Visualization of Very Large Graphs, G. Goos, J. Hartmanis and J. van Leeuwen (eds.) 1997. Graph drawing, Proceedings of the 5th International Symposium, GD'97, held in Rome, Italy, Sept. 18-20, 1997. Springer-Verlag, Berlin, pp. 403-414.
8. Leeuwen, J. van, Modelling architectural Design Information by Features, Eindhoven University of Technology, Eindhoven (1999)
9. Lieberman H. (1994) Powers of Ten Thousand: Navigating in Large Information Spaces. Proceedings of the ACM Symposium on User Interface Software and Technology, 1994. p.15-16
10. M.K.D. Coomans and H.H. Achten (1998) Mixed Task Domain Representation in VR-DIS. Proceedings of the 3rd Asia-Pacific Conference on Human Computer Interaction, APCHI'98, Shonan village Center, Japan, July 15-17, 1998.
11. B. de Vries, "VR-DIS research program", on the web site of the Design Systems group, http://www.ds.arch/ , 1997