

# A graph-based interface to complex hypermedia structure visualization

Manuel Freire  
EPS-Universidad Autónoma de Madrid  
Ctra. Colmenar Viejo km. 15,  
ES-28049 Madrid, Spain  
manuel.freire@ii.uam.es

Pilar Rodríguez  
EPS-Universidad Autónoma de Madrid  
Ctra. Colmenar Viejo km. 15,  
ES-28049 Madrid, Spain  
pilar.rodriguez@ii.uam.es

## ABSTRACT

Complex hypermedia structures can be difficult to author and maintain, especially when the usual hierarchic representation cannot capture important relations. We propose a graph-based direct manipulation interface that uses multiple focus+context techniques to avoid display clutter and information overload. A semantical fisheye lens based on hierarchical clustering allows the user to work on high-level abstracts of the structure. Navigation through the resulting graph is animated in order to avoid loss of orientation, with a force-directed algorithm in charge of generating successive layouts. Multiple views can be generated over the same data, each with independent settings for filtering, clustering and degree of zoom.

While these techniques are all well-known in the literature, it is their combination and application to the field of hypermedia authoring that constitutes a powerful tool for the development of next-generation hyperspaces.

A generic framework, CLOVER, and two specific applications for existing hypermedia systems have been implemented.

## Categories and Subject Descriptors

H.5.2 [User Interfaces]: [GUI, interaction styles]; H.5.4 [Hypertext/Hypermedia]: [architecture, navigation]

## General Terms

Algorithms, Design

## Keywords

Focus+context, graph visualization, hypermedia

## 1. MOTIVATION: A COMMON PROBLEM

The widespread availability of fast, interconnected computers has enabled the present boom of hypermedia informa-

tion systems. Despite recent progress in hypermedia authoring, advances such as adaptive hypermedia and reusable, modular contents are still far from widespread. In part, this is due to a lack of tools to create, edit and maintain the resulting structures beyond a certain size. This need is specially acute when non-technical users must create the contents.

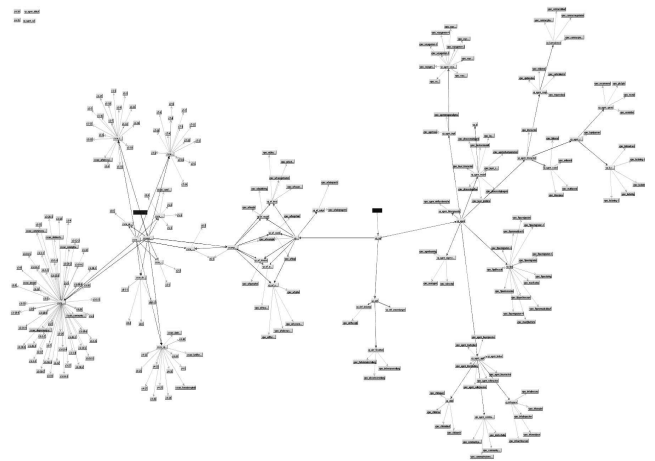


Figure 1: A first attempt at the visualization of complex hypermedia

Fig. 1 depicts the structure of repository in an xml-based reusable content management system. With this representation in hand, a careful observer would distinguish two entry points that share a small subset of the available material. The example was automatically generated from a Targeteam [18] repository.

Such a direct representation is of limited use: even though node count and connectivity are not very high, the resulting graph is much too large and complex to navigate effectively. However, the need for an easily understandable representation remains: without it, hyperspaces similar to the above are limited to what their creators and maintainers can plan and remember. The efficiency of development in hypermedia content is directly proportional to the degree of reuse that can be achieved, and this in turn is related to the size of the hyperspaces that can be managed.

The graph in Fig. 1 could have been dramatically simplified by abstracting away unnecessary detail. The definition of “unnecessary” is bound to change from case to case,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AVI 2004, Gallipoli (LE), Italy

Copyright 2004 ACM 1-58113-867-9/04/0500 ...\$5.00.

but an example would be that of Fig. 2, where the basic structure of the repository becomes evident. Color coding is used to distinguish the different node types, with darker nodes representing abstracted parts of the graph.

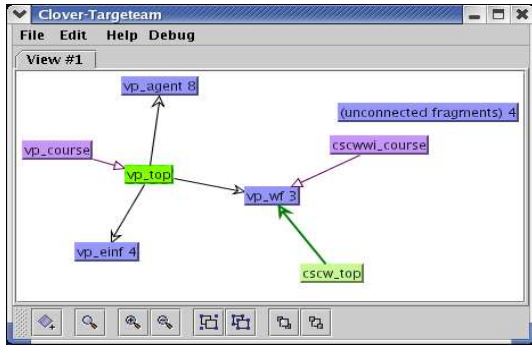


Figure 2: High-level abstraction of Fig. 1

The overview in Fig. 1 found in Fig. 2 was, again, automatically generated and laid out. It would be interesting to browse and edit the whole structure while keeping the graph complexity under control by means of similar overviews. To provide a direct manipulation interface that allows visualization and editing of abstracted hypermedia structures, a first requirement would be a means to *generate* the abstraction. This involves an analysis of the link structure, or even better, an analysis coupled with domain-dependent information (such as the fact that certain nodes should be always displayed, independently of the degree of abstraction). Secondly, it is necessary to provide means to *navigate* the graph, allowing the user control over the *degree of detail* in a manner that is both intuitive and flexible, so that neither orientation nor information is sacrificed. Finally, it should be possible to *modify* the structure by direct interaction on the interface.

The following section describes how the above requirements have been satisfied within the CLOVER framework. Some related work is examined in section 3, and finally conclusions and future plans can be found in section 4.

## 2. A CLUSTERING-ORIENTED APPROACH

We have developed a framework to experiment with the representation and editing of hypermedia structures. The name of the framework is CLOVER, an acronym that stands for *CLuster-Oriented Visualization EnviRonment*. It is the result of isolating the application-independent aspects of the system described in [9]. The framework provides visualization support for applications that wish to provide a graph-based interface to browse or edit hypermedia contents, such as the tool in Fig. 2.

In order to fulfill the goals stated in the previous section, hypermedia content undergoes a series of processing steps, depicted in Fig. 3. The first step is the *generation* of a graph from the appropriate part of the hypermedia system’s repository. The output of this stage is then *filtered* to exclude non-relevant graph fragments. In the *clustering* phase, a hierarchy is superimposed over the filtered graph. The higher levels of this hierarchy provide only rough overviews, while the lowest level contains the filtered graph in full detail. The hierarchy is used to generate abstracts that are then

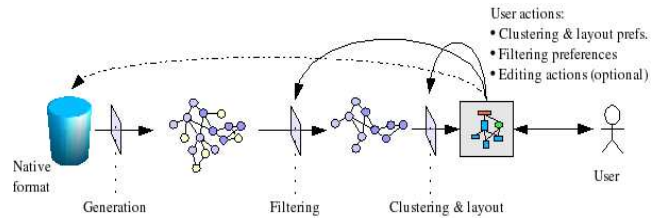


Figure 3: The CLOVER pipeline

presented to the user as an automatically laid-out graph. The user can affect the degree of detail of parts of the graph by means of a simple navigation scheme; modify parameters such as zoom, pan, and layout options, change the filtering (which requires all steps further ahead to be repeated), and modify the hypermedia space itself by editing contents or structure.

The core of the framework is the interaction between hierarchical clustering, detail selection, and layout—the last phase of Fig. 3. Clustering generates a multilevel overview; the exact level of detail is then chosen by use of a “semantic lens” algorithm, and finally the graph will be laid out with the chosen detail in the layout step. Because the user is free to navigate and edit the graph, clustering and level of detail will be updated frequently, which in turn forces a new layout to be undertaken. Therefore the final layout step includes dynamic elements to preserve user orientation.

Hierarchical clustering is performed by substituting “closely related” nodes for a single *cluster representative*, which can itself be used to form new clusters. The hypermedia structures we deal with have readily available semantic information, located in the contents of the nodes and in the different types of links. This, coupled with generally low graph connectivity, has motivated the choice of a simple, rule-based approach to clustering, easier to extend with domain knowledge than a general-purpose clustering algorithm. Applications using the framework can choose either to refine the algorithm to suit their particular domain (our approach up to date) or substitute it for a better fit. Clustering rules are presently executed in a series of iterations, where each node is examined, in a depth-first fashion and starting from the application-defined “roots”, to verify whether it matches with a clustering rule or not. Matching is done in reverse order, that is, from the leaves to the root. When a match is found for a node, the node will be elected as a representative, the future cluster is marked, and its nodes are no longer eligible for clustering during that iteration. The default rules guarantees that at least one cluster will be formed in each iteration, so worst-time performance is  $O(n^2)$ .

To allow efficient graph operations in hierarchically clustered graphs, cluster nodes have all incoming and outgoing edges found in their descendants. Multiple edges from one clustered node to another node (cluster or not) are collapsed into single *cluster edges*. For instance, if node  $v$  is connected to node  $w$ , all clusters containing  $v$  will have an outgoing edge to  $w$ —and to all clusters containing  $w$ . When operating on a slice, only edges between nodes in the slice are considered. This is a tradeoff between clustering speed and memory versus slice operation speed. A better implementation is suggested in [1].

## 2.1 Detail on demand with a semantic lens

Once the clustering is ready, some method is needed to decide what parts should be presented in high detail and which should be abstracted away. In information visualization, the term *fish-eye lens* was coined by Furnas [11] to refer to a technique of displaying a portion of the data with a greater degree of detail than its surroundings, operating both on the data and the way it is rendered. Furnas' original technique assigns a *degree of interest* (DoI) to information present in the display, and proceeds to hide or abstract those elements below a certain cutoff value. The DoI is affected by the *point of interest* (PoI), so that elements "nearer" to this point are assigned a higher interest. Refinements on this technique assign multiple types of DoI to each element, so that different aspects of the elements can be (un)highlighted independently. To distinguish it from techniques that only modify the representation, Furnas' method can be termed a *semantical fish-eye lens*.

Our implementation of Furnas' fish-eye lens uses the clustering hierarchy of the previous step and a simple distance metric (number of hops in shortest path to the PoI) to calculate DoI for all visible clusters. DoI for higher level (non-visible) clusters is defined recursively as the average DoI of their component nodes. When the graph is displayed, nodes that are near enough to the current PoI will be expanded. Those that are further away will be collapsed. The algorithm to select a new *visible set* is the following:

1. If the PoI is a newly selected cluster, expand it
2. Collapse everything except the "minimal set"
3. Until max. node number is reached,
  - a) Calculate DoI of non-expanded clusters
  - b) Expand the one with highest DoI

The *minimal set* is the combination of the PoI itself, a configurable (generally small) neighborhood of the PoI, and whichever nodes the user has previously marked as *frozen*. Frozen nodes will not be collapsed, expanded or moved from their present place in the layout, and allow the user to setup landmarks during navigation; the frozen state of a node can be toggled on or off at any moment via point-and-click.

The algorithm results in a series of collapses and expansions of the current visible set that, when carried out, yields a new set of visible nodes. *Visible sets* are also referred to as *slices*, for if the cluster hierarchy is thought of as a tree, in any moment the set of visible nodes would separate a portion of subsumed nodes (not visible because they are inside clusters) from another portion of clusters (too general to be represented). In graph theoretic terms, a slice is a *cut-set* of the clustering hierarchy tree.

## 2.2 Layout and animation of a clustered graph

Once the visible set has been selected, it is time to present it on the screen. A graph that makes use of a fish-eye (or any other multiple-detail) approach will change as the user's focus/PoI changes (by selecting a new focus with the mouse pointer) or editing actions take place. A smooth transition from one state to the next is necessary in order to *preserve the mental map* [8], and let the users focus on the change instead of having to re-understand the whole graph.

To provide orientation during changes of focus, areas undergoing changes are highlighted to draw the user's attention while the change is being animated. In our case, when a cluster disappears, all nodes of the cluster shrink towards

the center cluster. If a cluster is expanded, all nodes start initially in the position of the center cluster, radiating slowly outwards. And finally, when a new cluster that was not there before is added, a random, free position that is near to the center of gravity of the whole graph is selected (this need may arise when filtering changes).

Animation is achieved by a producer-consumer model. The producer is a variation of the *force-directed layout* (FDL) algorithm, where edges receive weights so that certain edge types will tend to be shorter than others. The consumer is a timer associated to the displaying window. The timer is configurable, but at about 5 to 10 new frames per second a smooth transition can be achieved. Despite there being better algorithms to choose from (see [10]), we have only implemented simple node interpolation for the animations. The producer thread samples the progress of the layout periodically, and the then-current node positions are added to a queue. When this queue reaches a certain length, the producer pauses until the queue has been at least half-emptied by the consumer. Fast processors will then use only a few bursts of layout, preserving user interface responsiveness. An advantage of using a producer-consumer model is the relative ease of producing "synthetic" animations for display (cluster expansion/node collapse): it is enough to fill the queue with a series of intermediate node positions.

At present, nodes consist mostly of textual information (icons contribute only a small portion of the total node area). This makes them considerably more wide than tall, and a final post-processing step is performed before the layout is complete to avoid node overlap.

## 3. RELATED WORK

Sougata and Foley described many of the techniques used in CLOVER in [17], although their work is centered on web-based hypermedia and relies heavily on hierarchization to generate graph representations.

The field of hypermedia that the authors are most familiar with is that of *adaptive hypermedia* courses. In general, researchers working in this field are well aware of the expressive power of graphs, and graph representations have been extensively used in the literature to represent AH course structures, as in [3], [4], [5], [15]. But this representation is certainly not frequent in the authoring side. Most authoring tools rely on more traditional dialogues and trees to maintain and change the structure. Some systems, such as De Bra et. al's adaptive hypermedia course system, *AHA 2.0* [5] are beginning to move in the direction of graph-based editing tools.

Force directed layout algorithms make use of a physical model, such as modeling vertices as ideal points and edges as springs that obey Hooke's law (the original spring layout, first proposed in [6]). Starting from an initially random layout, forces in the model are repeatedly calculated, and gradient descent is used to search for an energy minimum. For a good introduction to graph drawing, see [19]. A classification of graph drawing methods can be found in [12].

A general approach to the problem of fitting great amount of information into a small medium such as a screen is to use *focus+context* or *distortion-oriented* techniques. They all address the inevitable tradeoffs between detailed information in certain areas of interest (focus) and the bigger picture (the context). According to the classification by [14], these techniques can be applied either to the graphical represen-

tation or to the abstract data (as in our “semantic” lens). Examples of non-semantic fisheye lens can be found in [16] and [13]. The latter uses hyperbolic spaces to allow huge representations to be fit in a limited space by reducing detail exponentially fast at the borders. When applied to graphs, hyperbolic representations are best used with broad trees, where interactions between distinct branches are non-existent. This is not the case in many types of hypermedia.

Eades and Huang discuss the application of a semantical fish-eye lens via hierarchical clustering in [7]; their article has been a major inspiration for the present work. A more algorithmic approach to hierarchically clustered graphs is described in [1].

## 4. CONCLUSIONS

CLOVER provides a framework for the development of graph-based hypermedia authoring systems. Abstracts of the hypermedia graphs are performed via hierarchic clustering, by means of a simple yet effective rule-based algorithm. A semantical fisheye lens is in charge of providing variable detail. The user can dynamically adjust the level of detail, and the automatically generated layout changes smoothly in an effort to preserve user orientation. The use of Focus+context raises the issue of graph animation and the problem of preserving user orientation [8].

The framework makes no assumptions on the nature of the hypermedia system being edited, so many system-specific details have to be provided by the application using the framework. In order to validate the hypothesis, two system-specific applications have been designed and implemented, one for the adaptive hypermedia e-learning system Tangow [2][3], and one for the previously mentioned reusable content system Targeteam. Both applications are undergoing the final stages of implementation, and we hope to begin comprehensive testing in the near future.

Future work includes support for smoother animations, collaborative authoring, graph annotation, and path highlighting. We are also considering extending the framework for use as a platform for monitoring and tutoring students in a virtual learning community.

## 5. ACKNOWLEDGMENTS

This work has been sponsored by the Spanish Ministry of Science and Technology (MCyT) with project code TIC2001-0685-C02-01.

## 6. REFERENCES

- [1] A. L. Buchsbaum and J. R. Westbrook. Maintaining hierarchical graph views. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 566–575. Society for Industrial and Applied Mathematics, 2000.
- [2] R. M. Carro, E. Pulido, and P. Rodriguez. Dynamic generation of adaptive Internet-based courses. *Journal of Network and Computer Applications*, 22(4):249–257, Oct. 1999.
- [3] R. M. Carro, E. Pulido, and P. Rodriguez. TANGOW: a Model for Internet Based Learning. *International Journal on Continuing Education and Life-Long Learning*, 11(1–2), 2001.
- [4] A. Cristea and L. Aroyo. Adaptive authoring of adaptive educational hypermedia. *Lecture Notes in Computer Science*, 2347:122–132, 2002.
- [5] P. De Bra, A. Aerts, D. Smits, and N. Stash. AHA! Version 2.0, More Adaptation Flexibility for Authors. In *Proceedings of the AACE ELearn’2002 conference*, pages 240–246, Oct. 2002.
- [6] P. Eades. A heuristic for graph drawing. *Cong. Numer.*, 42:149–160, 1984.
- [7] P. Eades and M. L. Huang. Navigating clustered graphs using force-directed methods. *J. Graph Algorithms and Applications: Special Issue on Selected Papers from 1998 Symp. Graph Drawing*, 4(3):157–181, 2000.
- [8] P. Eades, Wei Lai, K. Misue, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, 6:183–210, 1995.
- [9] M. Freire and P. Rodriguez. Visualizacion basada en grafos para cursos hipermedia adaptativos. In *Proceedings of AiPO 2003*, 2003.
- [10] C. Friedrich and P. Eades. Graph drawing in motion. *Journal of Graph Algorithms and Applications*, 6(3):353–370, 2002.
- [11] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23. ACM Press, 1986.
- [12] I. Herman, G. Melançon, and M. S. Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.
- [13] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408. ACM Press/Addison-Wesley Publishing Co., 1995.
- [14] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. Comput.-Hum. Interact.*, 1(2):126–160, 1994.
- [15] D. B. Lowe, A. Ginige, M. Sifer, and J. Potter. The Matilda Data Model and its implications. In *Proceedings of the 3rd International Conference on Multimedia Modeling*, Nov. 1996.
- [16] M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss. Stretching the rubber sheet: A metaphor for viewing large layouts on small screens. In *Proceedings of the 6th Annual Symposium on User Interface Software and Technology*, pages 81–92, New York, NY, USA, Nov. 1993. ACM Press.
- [17] Sougata Mukherjea and James D. Foley. Visualizing the World-Wide Web with the navigational view builder. *Computer networks and ISDN Systems*, 27(6):1075–1087, 1995.
- [18] G. Teege. Reuse of teaching materials in Targeteam. In *International Workshop on Interactive Computer aided Learning ICL 2002*, 2002.
- [19] I. Tollis, P. Di Battista, P. Eades, and R. Tamassia. *Graph Drawing: Algorithms for the visualization of graphs*. Prentice Hall, 1998.