An Interactive System Based on Semantic Graphs

Johann Vandromme, Samuel Degrande, Patricia Plénacoste, Christophe Chaillou

An Interactive System Based on Semantic Graphs

Johann Vandromme, Samuel Degrande, Patricia Plénacoste,
and Christophe Chaillou

LIFL, Université de Lille 1, Cité scientifique
Bâtiment IRCICA, 50 Avenue Halley, Haute Borne, 59650 Villeneuve d'Ascq, France
{johann.vandromme,samuel.degrande,patricia.plenacoste,
christophe.chaillou}@lifl.fr

Abstract. This paper introduces an original method towards post-WIMP desktops. Our main contribution involves a new way of organizing desktops with respect of the Instrumental Interaction model and relies on semantic graphs. A semantic graph expresses semantic dependencies of documents or components like a scene graph expresses geometrical dependency. This structure allows users to visualize the desktop given a chosen criterion. We also propose polymorphic tools that enable Direct Manipulation: tools directly modify the semantic visual factor leading to a semantic modification of object. A prototype is proposed to illustrate this concept.

Keywords: software design, interfaces, graphs, semantic, desktop, tools.

1 Introduction

In WIMP systems, application functionalities are represented by buttons grouped in menus, sub-menus and tool boxes. Given the increasing number of these functionalities, graphical interfaces grow in complexity. Some rules and habits exist to design menus and interfaces but they are limited to common operations such as Open File, Save File, Print, etc. For specific tools, interaction modalities and their location in the interface may vary from one application to another (e.g. layer manipulation in 2D image editing software).

The second element that induces such complexity is the lack of interoperability between applications. To edit a specific type of file, one needs to launch a specialized application, then export the file in a generic format. The generic format can eventually be imported into another application. This export/import operation tends to be replaced by drag-and-drop between applications but is not generalized at the moment.

Current desktop managers and tools tend to provide features that are document-centered. The file-tree is about to disappear for indexation systems based on Meta data (WinFS[1], spotLight[2], beagle[3], google desktop[4]). The documents are linked in

[1] http://msdn.microsoft.com/en-us/magazine/cc164028.aspx
[2] http://support.apple.com/kb/HT2531?viewlocale=en_US
[3] http://beagle-project.org/
[4] http://desktop.google.com/features.html

several ways, using some semantic criterion. This increases the document research power that is not limited to document name, type, size or modification date.

Our first aim is to extend this principle to the documents content. We would like to bring uniformity in the desktop description with a generic structure for documents, components and system data.

Our second aim is to create a unique tool box for the whole system: each tool can act on every type of document. The main interest is to reduce the cognitive charge of the user by reusing polymorphic tools. In current systems, an application (tools and opened documents) are embedded into a single window. Given this unique toolbox, it will induce a window less system: the desktop is made of generic tools and a set of documents.

The third aim is to bind a visual parameter to a given semantic aspect. This allows a quick overview of documents sharing a common property and to extend direct manipulation by using graphical tools to modify semantic values.

2 State of Art

This section recaps previous works which introduce attempts to replace WIMP systems with direct manipulation. We will first present theory about instrumental interaction, then the software toolkits that implement this concept.

2.1 Direct Manipulation and Instrumental Interaction

In Shneiderman's direct manipulation definition [1], the user manipulates documents rather than files. This principle was first introduced by Xerox Star in 1981[2] and respects the three main principles of:

- Continuous representation of the objects and actions of interest: graphically modifying an object rather than using a dialog box;
- Physical actions or presses of labeled buttons instead of complex syntax: encourage graphical interfaces to command lines;
- Rapid incremental reversible operations whose effect on the object of interest is immediately visible.

These principles stimulate discovery and interface exploration and reduce the learning time by proposing intuitive interactions [3].

As an extension, Michel Beaudouin Lafon proposes the Instrumental Interaction model [4] in order to simplify current interfaces and to allow more flexibility for interaction development. The principle is to replace application-based interfaces by document-centered ones: Objects of the domain (i.e. documents) are made of a set of attributes that can be modified via interaction instruments: instruments are defined as attributes modifiers. As a complement for his model, he advises developers of interactive applications to make interaction as first class object and to separate the interaction code from the tool executive code [5]. These principles allow to create high level and post-wimp interactions, such as bi-manual ones.

Implementing high level interactions is difficult in current WIMP system. These 2D interfaces are built upon DOM trees [6] (e.g. XUL[5]). DOM trees are an equivalent of 2D scene graphs where each node is a wimp component (button, text box and layer) and expresses the geometrical dependence for a node to its parent. In such a structure, each component embeds its own interaction code as events. Therefore a high level interaction, or an interaction that needs a combination of different components (such as drag & drop), is split inside the DOM tree and becomes very hard to design or maintain; Designing interaction as first class object avoids the code of a high level interaction to be split among the nodes of a scene graph.

Beaudouin Lafon defined several principles for his model [7]:

- Reification is the process by which concepts are turned into objects – as an example: an interaction instrument is a reification of a command and instruments are objects that can be operated by Meta instruments.
- Reuse: an interaction or a set of interactions can be reused as pattern. For instance, the redo command can replace a set of interactions or the copy-paste allows reusing the result of previous user commands.
- Polymorphism is the property that enables a single command to be applicable on objects of different types. One can create polymorphic tool several ways: for current file format systems with multiple existing formats, it is necessary to create a tool with a code for each type of document. For a new type of document a new tool code has to be developed. The second possibility requires designing a common structure for each type of document. Thus, a tool can act on any type of document by modifying the common structure.

Instrumental interaction is a pertinent model oriented on direct manipulation. This is a formal model with several guidelines but without associated specification. Michel Beaudouin Lafon has tested it on a specific application dedicated to colored Petri nets [8]. This application includes a full set of modern interactions such as floating palettes, tool glasses and marking menus.

2.2 Related Toolkits

To achieve a system based on instrumental interaction model, toolkits will have to respect the separation of interaction with representation and introduce a generic structure for every type of documents for tool polymorphism.

Toolkits based on Instrumental Interaction rely on scene graph for representation, which is a classical design for 3D and now 2D graphical toolkits (like SVG[6] , or [9] [10]). This common model defines the relationship container/contained. However, programming interactions in such a graph is equivalent to current WIMP interaction definition.

To separate interaction from display, Huot & Dumas propose to design two different graphs: the DOM tree that deals with display and an interaction tree for the object's behavior [11]. In MaggLite based application, interaction graphs rely on ICON (Input Configurator) model [12]: the interaction graph is composed of devices. A

[5] http://developer.mozilla.org/en/XUL_Reference
[6] http://www.w3.org/TR/SVG

device is a black-box that can receive input values and can generate output values. The graph begin state is a system device (input peripheral), the internal states are a set of Library Devices (transform an input signal to get a modified one) and end states are Application devices (control the application objects). At the moment, this toolkit has been used to design classical windows-based applications and there is no proposition about managing a whole desktop system.

The DPI model (Document / presentation / interaction) [13] is an implementation model based on Michel Beaudouin Lafon's Instrumental Interaction. This generic model manipulates different types of document and can design a desktop system. The components of this model are documents (containing the domain data), Presentation (the representation of the document's data) and instruments (for interaction). The separation between presentation and documents allows multiple presentations coexistence.

An advantage of the DPI model is to provide a document structure to design polymorphic tools: Objects are defined as a set of attributes but unlike the Instrumental Interaction model, instruments don't modify directly attributes of documents: they create actions (tools are action producers) on components (action consumers). This system allows polymorphism: when an instrument tries to modify a document, it compares produced action (what the tool can do) to consumable action of the document (what is modifiable). If the actions are compatible, the object is modified. The DPI model is an important step in modeling a workspace centered on documents and instruments.

2.3 Limits

The previous systems present several drawbacks:

- Firstly tools are defined as attributes modifiers: they operate only on attributes and thus the structure of the geometric tree cannot be modified. The problem is, sometimes one needs to modify the scene graph (for instance, for a drag & drop operation). To solve this problem, one can define a specific tool that modifies the tree structure.
- Secondly these toolkits are dedicated to document edition. Their tools can only modify components, but cannot be used to manage documents.
- Finally interactions are based on graphical representations of documents. This principle limits the range of actions to graphical modifications. For instance, changing a geometrical aspect of a component using direct manipulation is possible but changing an attribute that expresses the relevance of a component often requires a dialog box.

3 Proposal

Our will is to extend the principles of instrumental interaction to manage a full desktop system. Our proposition must deal with components, documents or activities. Activities are several types of documents or components that share common semantic properties; to add semantic information, one classical possibility is to add attributes in each node in the DOM tree, but this principle doesn't deal with hierarchic semantic links. We propose to extend above toolkits capabilities by structuring the whole desktop using semantic multi-graph structures.

3.1 Structuring Desktop with Semantic Graphs

We have chosen to use several tree structures called semantic graphs. These structures will allow to group objects having hierarchical semantic properties (Fig 1).

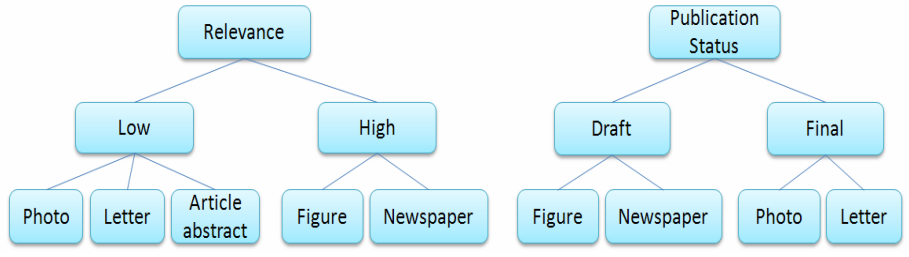

Fig. 1. Two examples of semantic graphs used in the desktop. On the left, the relevance graph groups documents depending on their relevance value: the figure and newspaper are highly relevant, the photo, letter and article abstract are not. On the right, the publication graph groups elements in two categories: draft or final. We can notice that the same objects can be present in different graphs.

A semantic graph expresses semantic dependencies of documents or components like a scene graph expresses geometrical dependency. It is defined this way: semantic properties are nodes and objects of the desktop are leafs. The objects of the desktop are documents or components (parts of documents). This definition allows to add granularity or precision in the semantic definition of an object by adding intermediate nodes in the graph. For instance, one can refine the relevance category by adding intermediate nodes that express the deadline time for each document.

According to that definition, a given object can be present in multiple graphs, given that an object can have different semantic properties.

The whole desktop is composed of semantic graphs:

- Some of them represent document structure. For example, the document's DOM-tree defines how components are assembled.
- Some are used by the system. As an example, document write-protection or the objects backups are stored as graph.
- Others can be created by the user to manage his/her activities: he/she can define his/ her own semantic values in order to group and retrieve objects the way he/she chooses.

3.2 Polymorphic Tools

Since the whole system is structured the same way, this leads to a harmonization of the desktop description and allows the tool polymorphism. The same tools can be used to modify a component (modifying an image), to manage the documents (positioning the image in the documents) or to manage the desktop (moving documents in the desktop): so, in our system, tools can modify objects' attributes (as in Instrumental Interaction) but also semantic links between objects.

These generic tools can be gathered in a unique tool box. So, an application defined as a container that gathers manageable documents and associated tools is no longer valid. It can be replaced to create a document centered system. Our desktop is made of a set of documents, a tool box and a set of graphs. Using the same tool box in a whole system allows transfer of knowledge when facing a new type of document: the user is familiar to his/her toolbox and knows the capabilities of the tools. Our tools respect interaction modality based on direct manipulation. In current applications, the user chooses the document to edit by opening it in an application: the choice of tools is defined by the application. In our system, one selects a tool and then applies it on a document. This modality is close to the real world where a user can take the same pen (thanks to polymorphism) to annotate a post-it, then to correct student's copies or to color a drawing.

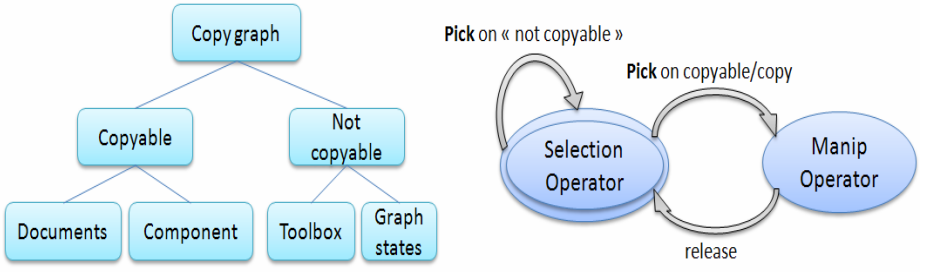

Fig. 2. An automaton that deals with the copy capability of an object: Combination of automaton rule and semantic graph won't allow copy of toolbox and graph state objects. The transition that loops on selection operator is defined via a rule.

Such a proposal (where a tool can act the same way with documents, components or system) is too permissive. First, some special actions must not be permitted in some cases. Secondly, some specific tools (tools that can be applied only on some types of documents) exist, so a mechanism is necessary to create exceptions. Our Tools are designed as interaction automata [14]: basic operators can be linked together to create a high level tool. A system of rules can modify the default behavior of automaton. Given a specific criterion, a rule can prevent a state change or can perform a specific action. As an example, there can be objects in the desktop system that cannot be copied (only one instance of the object is allowed). These objects can be defined as leaves for semantic parent “not copy-able”. A rule will avoid copy operation for all objects that share that property when a displacement operation is made (see Fig 2).

This system of rules will be used to apply specific behavior for one tool on one type of document or to manage the mechanism that deal with specific tools.

3.3 Direct Manipulation and Semantic Graphs

Our method allows direct manipulation extension. By associating a visual parameter to a semantic property, one can visualize a semantic aspect of the desktop. This system allows the user to display his documents from a chosen semantic point of view (Fig 3).

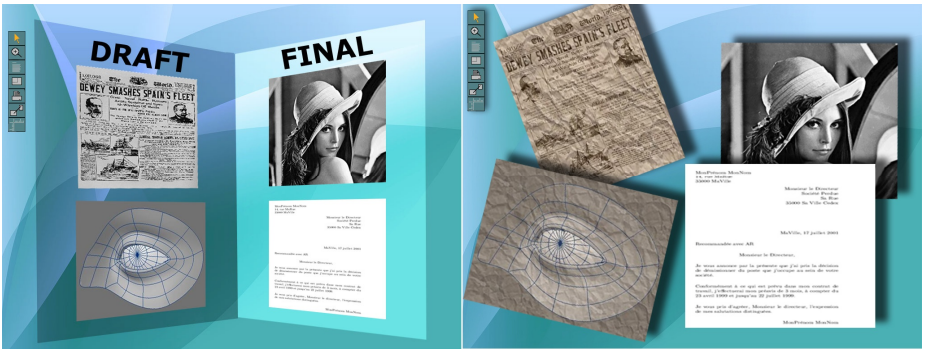

Fig. 3. A-B. Two different examples of semantic visualization of the desktop with the "publication status" semantic criteria: on the left, we use a labeled container for each value (draft or final). On the right, a crumple and rotation effect is applied to draft documents.

Fig. 4. The upper left desktop is visualized with "relevance" semantic value as scale visual factor. The upper right picture is displayed with blur effect as completion factor. The lower picture is an example of semantic composition of the two previous semantic values.

As a consequence, having a graphical representation of a semantic graph allows the use of generic graphical tools to change a semantic value of an object. This can enhance the use of direct manipulation and reduces the use of dialog-boxes by providing a way to use generic visual tools to modify semantic attributes. However, a "bijection" is

required between visual effect and semantic value for the visual modifications to be translated back to semantic values. In the example of the publication graph (Fig 3.A), moving a document from Draft to Final will modify its status.

A mechanism is necessary to select which semantic values will be set and how it will be visualized. It must be able to compose with different semantic properties to get a view that mixes several semantic values (Fig 4). This mechanism is independent of our semantic graph model, it depends only on the implementation of the model.

3.4 Predict User's Objectives

In specific cases, the system can help predict user's objective. Let us consider an object displacement from a container to another. A system solely based on the scene graph cannot determine whether the object is simply translated or a modification of the scene graph is required. The first case induces the modification of the position attribute and the latter case induces that the user wants to perform a drag-and-drop operation (the object hierarchy must change and induces to unlink the object from its parent to link it to the new container). A classical scene graph only provides the information that a shape has moved from one position to another. In our system, the system can obtain several semantic aspects of a component to determine what to do.

With a drag-and-drop operation in Fig 3.A, our solution can provide additional information to the system that a document has moved from a container "draft" to a container "final" which implies that the parent relationship must change.

4 Demonstration Desktop

To illustrate our proposition, we have developed a demonstration desktop based on semantic graphs. This desktop is document centered: it is made of a set of documents and a set of generic tools gathered in a unique toolbox. Several generic tools (which can act on different semantic aspects using direct manipulation) are proposed (relevance, publication, geometry...).

There are several possible strategies to manage specific tools. One can choose to make every tool available at any time. But this choice can enhance complexity of the tool box. Another possibility is to attach specific tools to specific documents, but this method is close to actual windowing systems that link documents to their tools via application context. We have chosen to manage with specific tools this way: specific tools of all displayed documents are added to the tool bar. This allows the user to have every tool for documents gathered and available and to keep the interaction modality used in real world: "choose the tool, then apply the tool". This is coupled with visual clues when a specific tool is activated to inform the user whether the tool can be applied on documents or not.

Our demonstration is based on OpenGL, but is still a 2D-desktop. Indeed, the 3D engine is used to display 3D effects, like overlapping elements using different depths cues. Another advantage of using a modern 3D engine is the possible use of extended visual effects. These effects can be obtained and modified in real time using Shaders. Our aim is to use these different visual effects such as lights, shades or blur to express semantic aspects of documents or elements.

For future works, we would like to integrate a Meta mode view. In this mode, the user can choose which graphs will be displayed from a list of semantic properties. The selection of several graphs will display every element that shares these properties. To apply a graphical effect to a semantic property, one can apply a visual tool directly on a selected node of a graph representation. The system will link this modification to the visual representation of every element that shares that property (Fig 5).

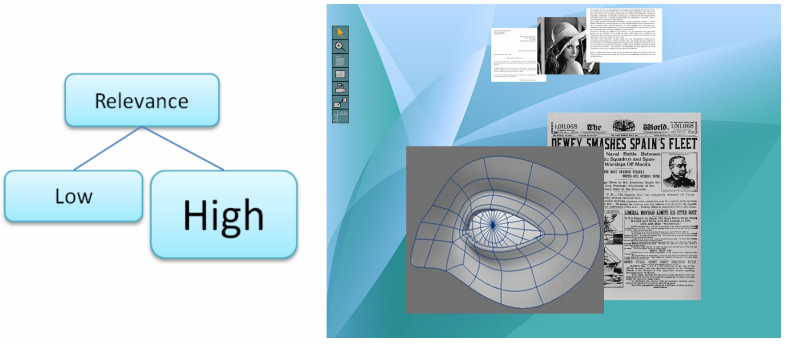

Fig. 5. In Meta mode, modifying the representation of the graph can modify associated objects: rescaling the "High" node in relevant graph (with the generic scale tool) rescales the associated objects on the desktop.

5 Conclusion

We propose a method for a desktop based on instrumental interaction with the use of semantic graphs. Semantic graph gathers documents or components that share common semantic values. Displaying a semantic graph allows to get different points of view of the desktop and to extend direct manipulation by using graphical tools to change a semantic value. The semantic graph structure is common to every component of the desktop system, thus enabling polymorphic tools creation. Every tool is gathered in a unique toolbox and a system of rules is set up to manage specific tools. A demonstration desktop is introduced as an implementation of our proposition. Several features are shown to prove our system capabilities. Future works will involve the creation of a Meta mode and evaluations to validate our approach.

References

1. Shneiderman, B.: Direct manipulation for comprehensible, predictable and controllable user interfaces. In: Moore, J., Edmonds, E., Puerta, A. (eds.) Proceedings of the 2nd international Conference on intelligent User interfaces, IUI 1997, Orlando, Florida, United States, January 6-9 (1997)
2. Johnson, J., Roberts, T.L., Verplank, W., Smith, D.C., Irby, C.H., Beard, M., Mackey, K.: The Xerox Star: A Retrospective. Computer 22(9), 11–26, 28–29 (1989)

3. Blanch, R.: PhD Thesis: Architecture logicielle et outils pour les interfaces hommes-machines graphiques avancées, http://en.scientificcommons.org/17563015
4. Beaudouin-Lafon, M.: Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2000, The Hague, The Netherlands, April 1-6 (2000)
5. Beaudouin-Lafon, M.: Designing interaction, not Interfaces. In: Proceedings of the working conference on Advanced visual interfaces, Gallipoli, Italy (2004)
6. Document Object Model (DOM) Level 2 Core Specification - W3C Recommendation November 13 (2000), http://www.w3.org/DOM/
7. Beaudouin-Lafon, M., Mackay, W.E.: Reification, polymorphism and reuse: three principles for designing visual interfaces. In: Proceedings of the Working Conference on Advanced Visual interfaces, AVI 2000, Palermo, Italy (2000)
8. Beaudouin-Lafon, M., et al.: CPN/Tools: A Post-WIMP Interface for Editing and Simulating Coloured Petri Nets. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 71–80. Springer, Heidelberg (2001)
9. Bederson, B., Meyer, J., Good, L., Jazz, L.: An extensible Zoomable User Interface Graphics Toolkit in Java. In: Proceedings of IUST 2000: 13th ACM Symposium on User Interface and Software Technology, ACM/SIGCHI, San Diego, CA, November 2000, pp. 171–180 (2000)
10. Lecolinet, E.: A molecular architecture for creating advanced interfaces. In: CHI Letters, pp. 135–144. ACM Press, New York (2003)
11. Huot, S., Dumas, C., Dragicevic, P., Fekete, J.D., Hégron, G.: The MaggLite post-WIMP toolkit: draw it, connect it and run it. In: Proceedings of the 17th Annual ACM Symposium on User interface Software and Technology, UIST 2004, Santa Fe, NM, USA, October 24-27 (2004)
12. Dragicevic, P., Fekete, J.D.: Input Device Selection and Interaction Configuration with ICON. In: Proceedings of IHM-HCI 2001, People and Computers XV – Interaction without Frontiers, Lille, France, September 2001, pp. 543–548. Springer, Heidelberg (2001)
13. Beaudoux, O., Beaudouin Lafon, M.: OpenDPI: a toolkit for developing Document-Centered environments. In: Actes Neuvièmes Journées sur l'Interaction Homme-Machine, IHM 1997, Poitiers, septembre 1997, Cépaduès-Editions (1997)
14. Martin, N., Degrande, S., Chaillou, C.: Adaptation dynamique de modalités d'interaction 3D par règles de filtrage. In: Journées de l'Association Français de Réalité Virtuelle (AFRV) (2006)