I N/JOY - The World of Objects

Manfred Tscheligi, Franz Penz, Martina Manhartsberger

Vienna User Interface Group
Department of Applied Computer Science
University of Vienna
Lenaugasse 2/8, A-1080 Wien, Austria
EARN/BITNET: A4424DAF @ AWIUNI11

Abstract

The first generation of graphical user interfaces can be improved by the application of visual design principles. These principles establish a new design philosophy, contributing to visual and intuitive human computer interfaces. Based on these principles we designed the visual interaction language of the integrated software system N/JOY for easy identification, manipulation and extension. The real life look and feel supports the user in the direct manipulation of normally hidden virtual concepts. System functionalities are designed and selected according to real life counterparts. The whole system is composed of parameterizable objects, avoiding cumbersome menu hierarchies.

1 Principles

The use of visual information in the human computer interaction now has been widely accepted as a means to reach a broader communication bandwidth between the user and the lot of possible system functionalities. The visual realization of submitting the intentions of the user to the computer system and the presentation of information from the system to the user can be designated as visual interaction language (VIL). This interpretation covers the user interface problem in general, contrary to existing classifications in [1, 2, 3]. They aim at characterizing visual programming and visual languages in the context of programming. A VIL on the other hand is the language used in the interactive bidirectional communication between the communication partners user and computer.

In this paper, we describe the underlying design principles of the VIL of an integrated software system called N/JOY that realizes the coexistence of wordprocessing-, graphics-, database- and spreadsheet functions. We tried to extend the user interface features known from the first generation of graphical interfaces to reach a second step of evolution in the look and feel of graphical interfaces. The alternative look and feel approach described in this paper offers the possibility to operate a software system in a more intuitive manner than possible with conventional interaction styles.

Our system can be characterized by the following short statements:

- Visualization of all system concepts transferred to the user (the prerequisite of direct manipulation)
- Systematic usage of a real life metaphor during development
- Behavior of all system objects according to the user's expectations
- Consistent syntax for operations
- Easy configurability according to the needs of the user

The consideration of all these principles results in a real alternative for the presentation of the complex system features found in modern software environments. In the following sections we give a detailed discussion of the principles. A companion paper [4] deals with the design concepts and the design methodology used to develop the user interface of N/JOY. A treatise of our VIL can be found in [5].

2 Visualizations of all system concepts

The notion of direct manipulation is often used to characterize modern, graphical user interfaces. Though several authors [6, 7, 8] tried to give a feeling for what direct manipulation is, an expressive and accepted definition is
missing. As an important prerequisite one can identify the explicit visualization of all system concepts (in contrast to their only virtual existence in non direct manipulative interfaces). Objects and their attributes are presented explicitly and can be manipulated by means of this presentation (I/O-identity). All user manipulations are fed back to the user immediately (feedback support). Object attributes are presented as part of the object presentation (presentational unity).

As a consequence we designed N/JOY to be completely composed of various objects, realizing the different functional features of the system. This results in an explicitly represented model world, where the user manipulates mentally as well as physically well defined objects with manipulation primitives forming a visual interaction language. These objects are not only passive data objects that can be manipulated via some menu initiated functions. All functions are integrated into the model world as well and presented to the user as objects having equal rights. This approach supports the engagement criteria mentioned by Hutchins [7]. The user does not make a distinction between a manipulated world and a manipulator world (usually the menu bar).

We omit global menu bars, a feature usually external to a manipulated world. The user is freed from searching through multiple layers of menus. In particular imperative commands have been abolished. Sometimes we used popup menus, but only if we could not find suitable visualizations for all object attributes in the object presentation itself. Nevertheless, in our system all system functions are integrated in one model world and accessible through the same manipulation primitives.

Objects fall into three general classes. Container objects are used to organize the environment. These objects contain other objects. For example, the filing cabinet, the folder or the drawer are container objects. A container merely influences the location of objects. Another class of objects, the functional objects initiate specific changes immediately. A copier copies other objects, a trash can deletes them. Data objects are the third class of objects. The document is the main data object. Marked text or a cell in a spreadsheet is also treated as an object.

A working environment (adjusted to typical office work) may contain a cabinet, some folders, a card-file enabling database functions, a trash can, a printer, a copier, a phone for direct communication with other users in a network environment, some pens, pads with printed forms, a calculator, a calendar etc. All functions are immediately available at any time merely by selecting the needed objects.

A typical example of the externalization of usually virtual concepts is the pencil object. We introduced a pencil object to define and use a special set of character attributes. Several pencil objects are possible, the user selects the one that is currently used. The user writes with the same attributes as long as the same pencil is used (this behavior is an alternative to the behavior in most systems, where the user inserts text with the attributes of a character close to the text insertion position). Another improvement is the explicit visualization of paste buffers. If the paste buffer is not explicitly presented, no direct manipulation is possible. In our system selected text parts can be manipulated directly as text modules, which are explicitly visualized objects. These text modules can be inserted, copied, deleted or printed, as other objects can.

Every manipulation needs a manipulator. We therefore used the concept of a hand object, which acts as a lengthening of the user hand. The hand object corresponds to the cursor presentation and effects all manipulations on the objects. To write with a special pencil, the user takes the pencil in the hand. The hand object is also a good place to collect some objects and to execute operations on more than one object (deleting some folders for example).

The user defines the available objects as operands or operators by applying different primitives of the VIL. The system uses this information to simply change an object state (e.g. a previously closed folder is opened) or to initiate more complex semantic actions. The action associated with the operator object is executed on one or more operand objects. It is this interaction relation between operator and operand which determines the dynamic behavior of the system.

3 The Real Life Metaphor

The visualization of system concepts comprises the question of how the information is presented. Direct manipulation requires the application of metaphors to map the functionality of the system onto objects, attributes of objects and relations among objects. Our system is designed according to a real life metaphor. In RLL-Uls (Real-Life-Look-User-Interfaces) the system functionality is presented to the user by elements familiar from daily life [9]. The appearance of the system elements imitates the real life model, transforming the concepts of the real life metaphor to a specific user interface instance. The real life acts as a special and well known system realization (metaphor) supporting the formation of a mental model for a new system. RLL-Uls are user interfaces primarily composed of real life looking objects.

The real life metaphor can be seen in some way as a generalization of the well-known desktop metaphor, extending it to the whole office environment and further to other objects of daily life. RLL is founded on the principle of simulation which signifies that the structure of a system realization should be close to the problem structure. We
also use the structure of the problem for the construction of the user interface.

Real life facets were already used by [10, 11]. Both systems do not proclaim a general user interface philosophy. They only try to simplify some of the features of their systems for the user. In our system, the real life concept is the leading look and feel principle. All design decisions are oriented at the real life look.

Subsequently we give a short scenario of entering N/JOY. Thereby we use terms of the intended conceptual model of the user. This mental imagination has to be transported from the user interface designer to the user by means of the presentations in the user interface.

Starting the application the user first has to enter a building containing an office by opening the main door (figure 1). Thereby an identification password may be asked corresponding to the key of the door. An automobile in front of the building would be a possible icon for leaving the application - driving home. Behind the entrance the user enters a corridor. Among other things it contains some further doors, one is the exit leading out of the building (figure 2). There is nothing uncommon in this corridor, it is just a room in the building as every further room behind one of the existing doors. In our system the user is enabled to build in rooms and doors connecting rooms as desired (if a new door from one room to a target room is built in this certainly results in a door from the target room to the starting room).

A typical working situation (room) with its inventory is shown in figure 3. The user has entered the room through the door in the upper right corner and can also leave the working room through this door. Below the door lies the object catalog which contains all objects available in the system. The user can choose the favorite inventory by ordering objects from this catalog. The drawer below the catalog may contain various tools like pencils or rulers. The printer, copier and trash can supply the corresponding functionality. On the left side we find some pads with the stationery needed, some pencils, and a filing cabinet. In the center of the room the user manipulates a document which remained open from the last working session. In the document a bar of the bar-chart is manipulated directly.

A pencil defines a special set of character attributes. The available pencils are lying on the users desk (figure 3). The activated pencil in use is inclined to show its working situation. Attributes attached to a pencil are easily identified by the appearance of the pencil. The form and size of the pencil visualization correspond to the selected attributes.

Each pencil is labeled adequately to provide an example text and the font information.

The real life metaphor is applied very early in the design phase. Starting with the planned system functionality we pragmatically distinguish between four categories of features. The classification is done according to the possibility of a RLL presentation (table 1).
In cases one and two planned features are implemented either supporting the real life metaphor or not. Copying a door with a copier object causes some troubles in real life. However, copying possibilities are a very important feature in such systems and therefore the interaction relationship providing this feature has to be allowed. If you copy a door, you get a new door leading into the same room.

In case three projected features are rejected to retain the real life feeling for the user. Usually a title bar is attached to a document window. We omitted the title bar because no real life document has one. A title bar only wastes valuable screen space. Dragging a document can be realized in a less wasting way. RLL objects need not be identified by an assigned name. The more common way to address objects with RLL is by their position (via a pointing device), form, color or simply by their contents.

Finally in case four features are added that originally were not intended but support the real life feeling. The possibility to take something out of the trash can was not planned initially (figure 4). But as we found a good presentation, i.e. taking the object out of the trash can by a visible remainder of the deleted object, we included this feature.

To give the user a real-life feeling real life presentations have to be provided. At the current stage of the system a first level of recognizability and virtual reality has been achieved. The images are constrained by the software and hardware environment used. All real life objects need an improved quality to achieve a sophisticated level of user satisfaction. Therefore the implementation and use of different real life objects should be supported by powerful multidimensional computer graphics concepts. The production of graphical user interfaces is computer graphics programming.

We faced some problems regarding the transformation of real life ideas into the system. There is e.g. the problem to find appropriate visualizations supporting the identification and distinction. The sizes of objects have to be different from the sizes of the real life counterparts. We tried to keep the relations of sizes to intensify the real-life feeling for the user. Sometimes misinterpretations occur due to the object position. There are no restrictions in the position of the objects as in real life. This differs from the real life metaphor but has the advantage of increasing the user comfort. In spite of these problems the extensive use of real life concepts offers a valid design alternative. For every system feature needed, we found an appropriate visualization. The specification document is easily understandable for the programmer and can be transformed into the final program without considerable consultation of the user interface architects.

### 4 Behavior according to the user's expectations

The identity of the user's expectations of a system reaction and the real, observed system reaction can be used as a 'correctness condition' for user interfaces. During our design we always asked 'What will the user expect to happen?'. The real life look supports a design process by making it easy to guess the right answer. The behavior of the various objects is strongly associated with their real life appearance.

In the terms of Hutchins [7] this supports the reduction of semantic distance. Intentions of the users result from their experience with other systems. The most important system is the real life. If we adjust the behavior of our system to a real life behavior this level of distance between user and system is influenced. Sometimes it is even necessary to reject preliminary planned system functions because of the difficulty in finding an appropriate presentation.

What happens in a system has to be identical with the user's anticipation based on the presentations in the user interface. This extends the WYSIWYG principle, originated from the possibility of a one-to-one mapping between documents visible on the computer screen and their output on paper to a What You Believe Is What You Get (WYBIWYG) approach. This extended approach also influences the functionality of a product and therefore the whole development cycle.

In this regard the notion of modes gets a somewhat different interpretation. Usually modes are understood as dis-

<table>
<thead>
<tr>
<th>case</th>
<th>planned</th>
<th>presentation in RLL possible</th>
<th>feature implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 1 Classification of planned features

Figure 4 Taking an object out of the trash can
tinct system states, where different subsets of system operations are executable. Modes, defined in this traditional way, are hard to avoid in practice. In our understanding, a mode is harmful only if the user expects specific system features in a particular state and the system does not behave in the expected manner. This belief results from the user's experience in real life, not only from the system behavior observed.

5 Consistent syntax

The N/JOY user has to remember only a few interaction primitives to manipulate the objects and to utilize their assigned functionality. These primitives can be applied to all objects.

In N/JOY the interaction from the user to the system is performed visually by manipulations of the objects available. Four manipulation primitives form a simple visual interaction language.

At the semantic level the mouse is the lengthening of the user's hand into the system. The user can take an object into/out of the hand (mark/unmark) and initiate (activate) the individual functionality of an object. The functionality is applied to the objects in the hand. The activated object acts as an operator which is applied to the marked operands in the user's hand.

This operator/operand relation is symmetric. E.g. a copier can be deleted by means of a trash can and the trash can can be copied by means of the copier. An interaction does not make sense for every pair of objects but there are a lot of objects which carry functionality making sense for most objects.

Further the user can open/close an object to view/hide the contents of the object and open/close an object specific dialog box to access the various parameters of the object.

The syntactical level is characterized by no restriction regarding the order of the interaction primitives (tokens): activate, mark, open, dialog. All valid interactions can formally be stated as:

\[
\{ \text{activate, mark, open, dialog } \}^* \]

The four tokens are lexically combined from a mouse movement to the target object, followed by one of four click actions. A user can touch an object in the following different ways: left/right, single/double click with the mouse.

A single left click on an object marks/demarks the object as operand for future operations. A single right click on an object activates the object functionality which is applied to the marked operands.

The left double click is used to open objects which may be opened (for example doors, cabinets, trash cans, folders etc.). The right double click opens an object specific dialog box which allows the adjustment of not directly touchable object attributes.

The mark/activate interaction can also be performed as usual in most existing direct manipulative systems by dragging the operand object to the operator object. Yet this method seems to be less appreciated by the users, perhaps it is uncomfortable to hold down a finger pressing a mouse button for a longer period.

The use of few and consistent interaction primitives supports the input component of the second level of distance mentioned by Hutchins in [7]. All objects are manipulated in a direct and uniform way, independent of their specific semantic role. The deduction of the meaning at the semantic level has to be supported by high quality graphical presentations which exactly show the real life counterpart.

6 Easy configurability

The leading source for new objects in N/JOY is the catalog. The catalog contains all objects available in the system. The user can take objects in any number according to personal needs. If additional objects are introduced during later enhancements of the product, they can be integrated in an altered catalog. A catalog may also be adjusted to different user classes.

New object instances are dragged out of the catalog. Object instances can be parameterized through special value settings. Multiple instances of the same object type can exist simultaneously, providing the user with the power of parameterizable objects [12]. These objects are used as operators to initiate semantic actions, and therefore the user needs no menu adjustments.

As in real life, a typical user will have about four pencils lying on the desk defining favorite text attributes. The easy access to the prepared pencils leads to uniformity of the documents produced. In an institutional environment uniformity can also be achieved by supplying the same set of pencils to all members.

Our system is based on individual objects. Our approach is suitable to adapt several objects to different cultural societies or business standards. To switch over to a different context only some object presentations have to be altered. The basic semantic remains constant.

7 Conclusion

With the user interface of N/JOY we tried to close the gap to the user by means of the following important design principles: visualization of all system concepts, real life metaphor, behavior according to the user's expectations, consistent syntax, integration of functions, and easy
configuration. The difference to other products is not achieved by totally new ideas, but by the consequent application of direct manipulation and WYSIWYG leading to the principle of WYBIWYG.

The first version of NJOY now shipped includes the most important features of our design. Yet from the user interface designer's point of view it is only the first prototype because a real rapid prototyping is not possible in the OS2/PM development environment. Some features which are not included now or which are not implemented in the manner designed will be added or altered in a future release. For example animating components like growing plant objects will give the user further enjoyment.

With the possibility of object parameterization some sort of programming features are integrated. To enhance the programming power we have to solve the problem of iteration, conditionals and variables within the general interaction philosophy to fulfill the definition of programming according to [3].

Currently we are also working on an adequate presentation of the database functionality and the application of the real life look interaction style to the technical domain of petri net editing.

References


