Cooperation between Reactive 3D Objects and a Multimodal X Window Kernel for CAD

Patrick Bourdot, Mike Krus, Rachid Gherbi

LIMSI-CNRS UPR 3251, University Paris XI, BP 133, 91403 Orsay cedex, France
mix3d@limsi.fr.

Abstract. From the early steps of sketching to final engineering, a frequent and very important activity in designing objects is to perform graphical and spatial simulations to solve the constraints on the objects which are being designed. But when we analyse work situations involving the use of CAD systems, it is today an acknowledged fact that these tools are not helpful to perform these types of simulations. While knowledge modeling based on *form feature* concepts already offers some possibilities for attaching behaviour to objects, the simulation activity requires in addition a 'real time' and 'intelligent' management of the interactions between the 3D virtual objects and the CAD user.

Our general purpose is to study how future CAD systems could be improved to achieve the simulation steps of object design. In this context we present some issues concerning the cooperation between a model of reactive 3D objects and a multimodal X Window kernel. We have developed a prototype of a system where objects with reactive behaviour can be built, and with which the user can interact with a combination of graphical actions and vocal commands. This prototype is used to evaluate the feasability and the usefulness of the integration of such techniques in futur applications that would be used by object designers in a real working context. We describe the current state of this system and the planned improvements.

1 Introduction

One of the most promising results of the research on multimodal user interfaces is their capacity to transform industrial applications which manipulate 2D or 3D virtual spaces. For example, it has been shown that a user can be more 'productive' using a CAD program when keyboard interactions are replaced by vocal ones (Martin, 1989). Hence some researchers propose to rethink these tools in terms of multimodal interaction (Gaildrat et al., 1993).

More than with simple substitutions of modalities added to the traditional user interface of CAD systems, our interest is to study how multimodal interaction could make these systems contribute better to object design. In fact, object design is not only an engineering activity, for which CAD systems are already useful. A frequent and very important activity is to perform graphical and spatial simulations to solve the constraints on the objects which are being

designed. Unfortunately, these simulations are not easily performed with current CAD systems, and the use of such tools for this activity requires complicated work organisations (Lebahar, 1992). We will hereafter refer to such simulations as 'object simulations' or '3D simulations'.

In this context, our general purpose is to study how future CAD systems could be really helpful for object simulation. Indeed, this activity requires a 'real time' and 'intelligent' management of the interactions between the 3D virtual objects and/or between these objects and the CAD user. An object (already built by the user or under construction) must have short-delay reactive behaviour in relation to the other objects of the virtual space it has dependencies with. Additionally, the user's interactions with these virtual objects must be efficient, requiring advanced combinations of modalities both for input and output, syntactic and semantic dialogue analysis, and so on.

But reactive behaviour as well as multimodal interaction with 3D virtual objects presuppose knowledge modeling for these objects. During the last ten years, knowledge modeling for CAD/CAM is one of the main purposes of the *form feature* approach. According to Shah (1990), "a *form feature* is a physical constituent of a part with a generic shape realisable or abstract, it has significance in design, analysis, manufacturing, or some other engineering domain and has predictable behaviour or properties". From this perspective, parametrical and variational modeling concepts of 'form feature' are very close to the requirements of a CAD system when used for 2D or 3D object simulations. On the other hand, we have to take into account that the simulation activity for object design can start from the first steps of sketching in the design process. In other words, during some simulation steps the objects might not have sufficient properties to be attached to form feature classes but only to geometrical semantics. Considering that 'good' knowledge modeling generally requires a hierarchical strategy, we chose to focus our work on the simulation activity with geometrical and topological objects. Indeed, our objective is to construct a 3D modeling kernel for object simulation that future CAD systems dedicated to particular object design domains could share.

However, manipulating virtual spaces requires a powerful graphical environment. Today, many of these applications are developed on UNIX workstations using X Window as a standard windowing environment, sometimes with additional graphical hardware. In spite of the various functions it can realise, the X server only manages, in terms of input modalities, mouse and keyboard events. In order to support the kind of interactions we require, we have created an architecture which seamlessly integrates new advanced modalities.

In the remainder of this chapter, we first define some basic concepts of multimodality and we show the importance of semantic representations for multimodal user interfaces. We subsequently present the geometrical and topological model of MIX 3D *(Multimodal Interactions under X environment with a 3D virtual space)* and we explain how knowledge modeling in MIX 3D is used to manage objects simulation. We then present examples of cooperation between reactive 3D objects and multimodal interactions. Finally, we discuss the design of our

multimodal interface architecture in the X Window environment and consider some general requirements for future enhancements.

2 Multimodal Interactions: Main Concepts

2.1 Definitions of Multimodality

Multimodal user interfaces are frequently opposed to multimedia applications (Bellik et al., 1995a; Schomaker et al., 1995). The task of a multimedia application is to allow users to manage large volumes of information represented by means of several different media (images, texts, sounds, etc.). The task of a multimodal user interface is at the input side to analyse, recognise and merge information coming from several devices, and on the output side to determine an appropriate combination of media to deliver a message.

Using distinctions of this kind, a three dimensional classification of multimedia and multimodal systems was introduced by Nigay and Coutaz (1993). One dimension describes the *levels of abstraction* of a system, while the two others represent respectively the *fusion levels* and the *use of modalities* that a system is able to support. For instance, a multimedia application uses low levels of abstraction because it does not take into account any semantics of the data to determine meaning. At the opposite end, a multimodal user interface has several levels of abstraction from raw data to symbolic representations. These representations allow the system to perform artificial reasoning and to improve the human-machine interaction. The 'fusion levels' dimension introduces a distinction between *independent* and *combined* user interfaces. The former are able to use one modality to receive or produce an expression, while the latter allow to build expressions from (resp. with) several input (resp. output) modalities. In the same way, the 'use of modalities' dimension has two possible values. A user interface is *sequential* or *parallel* if a single modality or several modalities are managed at the same time.

The possibility of combining several modalities which are chronologically or synchronously processed is thus an important characteristic of multimodal systems. In the following we study the fusion of input modalities. An input modality produces monomodal events. By hypothesis, a monomodal event can be the result of a recognition system associated with the modality's input device. Furthermore, we will see that the state of some devices and the state of their associated recognition systems are also useful for the fusion of modalities.

2.2 Temporal Proximity as a Fusion Criterion

The main concept of multimodal systems is the combination of monomodal events. It is necessary to define some criteria allowing us to decide how and when it is possible to achieve this combination. For instance, temporally close events in one or several modalities may have to be merged into the same multimodal expression. But the detection of this temporal closeness might not be sufficient

to perform a semantic interpretation. Below we will discuss some other criteria that can be used to supplement this *temporal proximity* criterion. Nevertheless, it is important to take this proximity in account. Indeed, in conjunction with a precise time stamping (or *'dating'*), this criterion allows us to overcome the problem of parallelism in monoprocessor systems in order to simulate *parallel* and *combined* multimodal interaction. More importantly, temporal proximity management seems well-adapted to the physiological behaviour of the human operator when he interacts with a multimodal system. Indeed, when a human uses more than one modality, the temporal synchronisation of his actions does not have the exactness of dating by the machine.

Temporal proximity is the main criterion for combining different modalities. For instance, the fusion of input events proposed by Bellik and Teil (1993) first determines the events which are produced within a short time range, and then combines some of them with respect to other criteria. With this fusion process, the classical multimodal expression *"put this here"* associated with two graphical selections becomes possible. The temporal proximity filter allows the determination of the co-references between the *"this"* and *"here"* vocal events, and the first and the second selection, respectively.

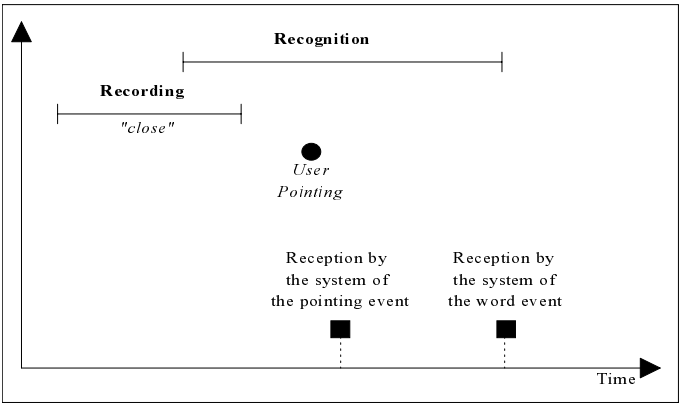

Fig. 1. The problem of device response times with speech recognition (taken from Bellik et al., 1995a).

A sequence of monomodal events can also produce different multimodal expressions, however, because the semantic interpretation can depend on the chronology of events. As explained by Bellik et al. (1995a), misinterpretations are possible at the fusion decision step (Fig. 1), since the devices associated with recognition processes (speech, gesture, ...) need much more time to analyse an expression than any standard input devices (mouse, keyboard, ...) or than any non-standard devices without recognition process (tactile screen, eye-tracker, ...). Hence, the only solution to avoid undesirable actions on the objects of the

multimodal application is to know exactly the starting date and the duration of the recognition process of each monomodal expression.

A precise time stamping is a difficult problem, because our management of non-standard devices has to be done within a UNIX / X Window environment (and the same probably holds for other environments). Section 5 of this chapter will show how we manage the temporal proximity criterion for the fusion of input modalities in such an environment.

2.3 Other Fusion Criteria and Semantic Information

We described above the concept of the temporal proximity fusion criterion. In order to achieve a complete and valid fusion, it is necessary to use additional criteria which provide more semantic information. We present here four fusion criteria introduced by Coutaz et al. (1992).

1. *logical complementarity*:
   This criterion allows the merging of temporally distant events within the same command. Still, the fusion's validity depends mainly on a value of this temporal distance. Indeed, when the user interacts with an object, the 'shorter' this distance, the more the fusion can be done at interface level. When it gets longer, it becomes very difficult to perform the fusion without the control of the application. So if we do not want to restrict the user's actions, it is necessary to implement a reactive object mechanism allowing the application to overcome such restrictions.
2. *data structure completeness*:
   Generally, this constitutes a condition to move within the *abstraction levels*. Usually, the generic levels concern the interface while the semantically complex levels are managed by the application. This completeness is also useful to stop waiting for other eventual events. Indeed, this criterion allows the system to decide if all the arguments of a command match with the data.
3. *dialogue context*:
   This criterion uses the historical log of the interactions. With this criterion it is possible on one hand, to determine the co-references between modalities (when a modality cannot be correctly understood without events or states from other modalities), and on the other hand, to handle anaphora, ellipsis and deictic expressions.
4. *incompatibility of modalities*:
   This criterion forbids the integration of modalities that cannot be used together. In particular, it detects contradictory monomodal expressions. While the interface can in most cases detect lexical, syntactic and semantical incompatibilities, some of these cannot be handled without knowledge of the application domain.

For most criteria it seems that some part depends on the user interface process while another requires interaction with the application:

$$Criteria(interaction) = Criteria(interface) \cup Criteria(application)$$

For the application, the validation of these criteria generally requires an analysis of complex semantic representations. Furthermore, for the 3D simulations required by object design, the access to these representations must be immediate to guarantee natural user actions and realistic object reactions. The next two sections present how the MIX 3D application controls and validates some of these criteria with the help of reactive objects.

3 A Semantic 3D Model for Object Simulations

CAD systems generally use a B-REP model (Boundary REPresentation) to describe solid objects. Historically, the first one was the Winged-Edge structure for polyhedron entities (Baumgardt, 1972), but now other data structures are existing which authorise quadric (Levin, 1980) or bicubic patches (Carlson, 1982; Casale, 1987). Although a B-REP accepting free-form surfaces is very interesting for design activities of complex objects, the resulting curves of intersecting surfaces have often must be computed by approximation methods. Consequently, this type of B-REP depends on the precision of the representation needed. As this precision is liable to change during the design process, a complementary data structure must save the modeling steps.

A solution is to use CSG trees (Constructive Solid Geometry), where the leaves are volumetric primitives, while the nodes are regular operators (Tilove and Requicha, 1980) combined with a geometric transformation or a shape deformation (Barr, 1984). But CSG trees are not well adapted to manage curves, surfaces or any free-form objects, because they were initially created to manage internal composition laws defined on a set of solid objects. On the other hand, though a CSG tree can be used to re-play the 3D modeling steps, it is clear that something else is necessary to give reactive behaviour to objects in order to make simulation activities possible during the design process.

For MIX 3D we have defined a model which combines knowledge modeling of geometry and topology with a generalisation of CSG trees. In the rest of this section we present these two aspects of our approach, followed by an example of the reactive behaviour of MIX 3D objects.

3.1 Knowledge Modeling for Geometrical and Topological Properties

Figure 2 presents an overview of the three typologies used in MIX 3D: the *geometrical* typology which represents the shape and the metric of 3D objects; the *fitting* typology which describes metric relationship properties between objects; and the *topological* typology which defines non-metric properties.

MIX 3D objects are identified primarily according to one of the generic semantic types of the geometrical typology: `Point`, `Curve`, `Surface`, `Volume`, `Structure` and `System`. By definition, a MIX 3D object is described as a set of data representations. Each representation is an instantiation of one structural

specialisation of its generic semantic type. We explain later how we manage these sets of data representations (section 3.2).

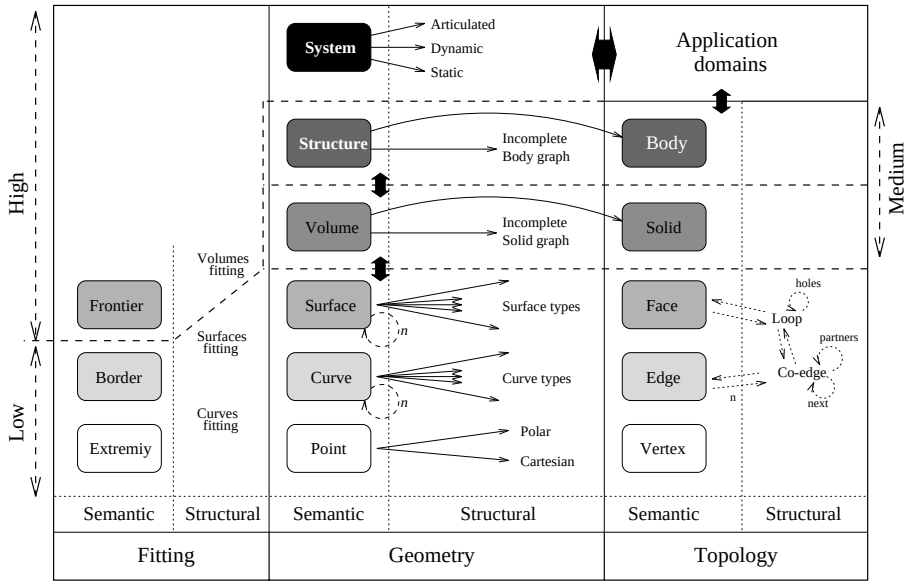

Fig. 2. The three typologies of MIX 3D.

Examples of geometrical knowledge According to Fig. 3 and the previous definition of MIX 3D objects, a `Curve` whose semantic aspects are `free` and `cubic` has two possible basic data representations: a `SplineCurve` object and a `Composite` object defined by `BézierCurve` objects. But it is also possible to give it more complex representations, by using the (possibly recursive) definition of the `Composite` type with `SplineCurve` elements, or `BézierCurve` elements, or both. In the same way, the data representation of a `Curve` which is semantically `free` but now `mixed` is a `Composite` object defined (recursively or not) by any `Parametric` object.

Besides the `planar` and `closed` aspects, a `polygon` has the same possible representations as a `linear free Curve`: a `PolyLine` or a `Composite` object defined (recursively or not) by `LineSegment` and/or `PolyLine` objects. But the default data representation of the `polygon` is a non-recursive `Composite` object defined by `LineSegment` components only, simply because this decomposition is useful to describe the fitting properties of a `polygon`. On the other hand, the default data representation of a `segment` is a `LineSegment`, but it may also have a `PolyLine` or a `SplineCurve` representation when this is useful for some 3D modeling operations. Additionally, the representation of a `conic Curve` can be

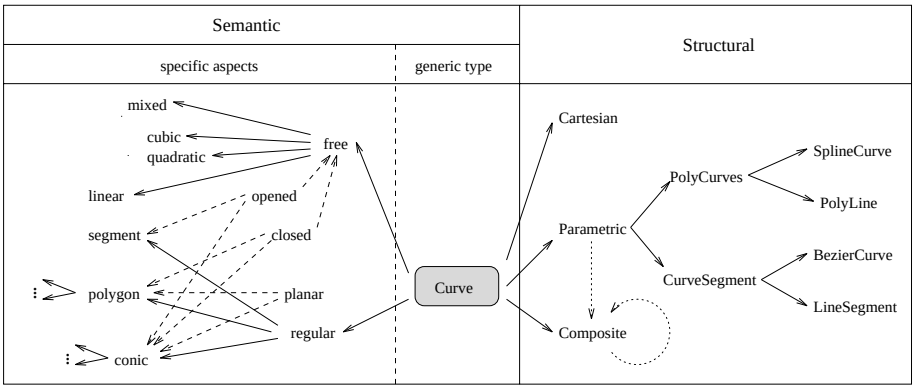

Fig. 3. `Curve` specialisations of the geometrical typology.

its `Cartesian` equation or a specific `SplineCurve` (a quadratic Non-Uniform Rational B-Spline).

These examples of the `Curve` specification clearly show that a similar semantic and structural classification must exist for the `Surface` type. Concerning the `Volume` and `Structure` generic geometrical types, no specific semantic aspects were identified. In fact, these two types of object are very important for the construction of the topological graphs, using some of the properties of the lower levels of the fitting and the geometrical typologies.

Fitting and topological knowledge An instantiation of a generic semantic class of the geometrical typology can be specialised with the help of the semantic class of the same level within topological or fitting typologies (Fig. 2). For instance, a `Curve` can describe the `Border` (fitting type) of a `Surface`. Moreover, if another `Border` of another `Surface` is described by exactly the same `Curve` and if these two `Surface` objects have a C^0 continuity, then this `Curve` is the metric of an `Edge` (topological type). In fact, this `Curve` is a kind of 'pivot' of the relationship which exists between the two `Surface` objects to make them fit with this specific property of continuity.

The fitting typology covers more general cases than simple C^0 continuity between two `Surface` objects. Figure 4 shows an example of all the structural relations that a `SurfaceFitting` object describes. Some properties (and constraints) that a `SurfaceFitting` object is presently supposed to support are for instance: α properties (angular relationship), β properties (i.e. the *shape parameters* introduced by De Rose and Barsky (1985) for *geometric continuity*), δ properties (linear or arc lengths), π properties (parallel relationship) and so on. Additionally, combinations of these properties define some macro-properties. For instance, an α property specifies a β property as a *continuity* if $\alpha = 180°$, as a *coincidence* if $\alpha = 0°$, or as an *incidence* for other values of α. On the other hand, a `SurfaceFitting` object can describe the fitting of $i \times j$ `Surface` objects

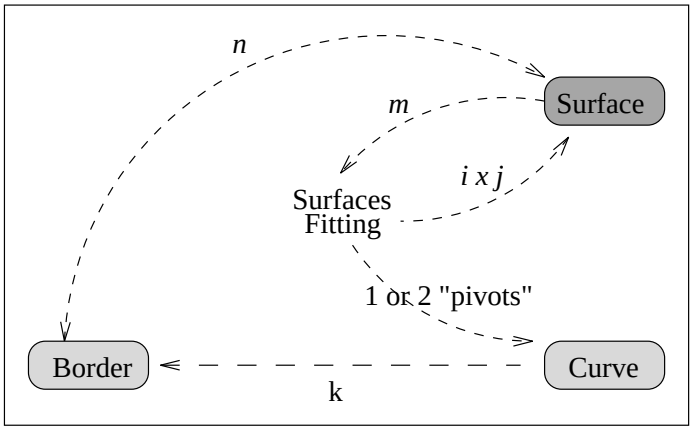

Fig. 4. Some classes of the fitting typology and their relations with the geometrical typology.

which is useful to build surfaces with a patchwork structure (Macé and Bourdot, 1991).

3.2 The Generalised Constructive Graph

One of the main principles which appeared in our previous discussion about knowledge modeling for MIX 3D objects is that the semantic aspects of a generic type of a typology is related to one or more primitives of its structural specialisation. Our goal is to allow the user to ignore the low-level data structures of a CAD application to let him manipulate the semantic aspects of geometry, fitting and topology only (a `Curve`, a `Surface`, their degrees, closures, fitting properties, their topological aspects and so on). So far we have only presented which knowledge we describe and how, but not how this knowledge is used during a 3D modeling tasks and spatial simulations. In MIX 3D, knowledge is managed by means of the Generalised Constructive Graph (GCG).

Definition of GCG nodes Two types of nodes of our GCG are structures called `Form (F)` and `Instantiated-Form (IF)` (see Fig.5). A node of the type `Instantiated-Form` is a MIX 3D object. Its spatial position is given by a *localisation matrix*. The modeling description of an `Instantiated-Form` is a set of data representations that a MIX 3D object may have according to the semantic aspects it has with respect to its generic geometrical type (see section 3.1 and Fig. 2). A `Form` node is a user class of MIX 3D objects. Any data representation of an `Instantiated-Form` is inherited from its `Form`. This is done according to the localisation matrix and the 3D modeling context in which this `Instantiated-Form` is used.

On the other hand, a `Form` results of the application of an *operator* to an ordered list of components. An operator is any 3D modeling method that MIX 3D

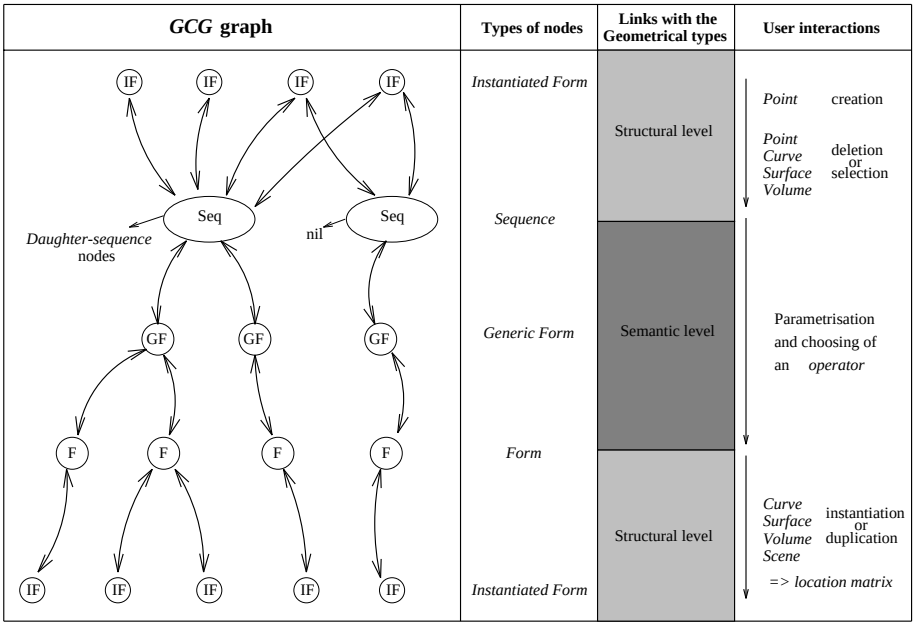

Fig. 5. The Generalised Constructive Graph used by MIX 3D.

objects can support, while each component of this ordered list is an existing `Instantiated-Form`. Because the `Instantiated-Form` nodes can be shared by several lists of components, we manage these lists with a third type of node called `Sequence (S)`. According to the knowledge carried by the GCG nodes, a `Sequence` under construction dynamically determines the list of *possible operators* which can be applied to its list of `Instantiated-Form` components.

Finally, a fourth type of node is the `Generic Form (GF)` structure, which represents a set of `Form` nodes sharing the same `Sequence` and having the same generic type according to the geometrical typology. It is through `Generic Form` nodes that the semantic aspects of a `Form` are managed. For instance, this node is used to determine, for an `Instantiated-Form` within a given `Sequence`, which data representation is useful to support the application of one of the possible operators this `Sequence` authorises.

Properties of the GCG `Instantiated-Form` nodes also have components, but the components of such nodes sharing the same `Sequence` generally don't have the same geometric localisation as the components of this `Sequence`. If the `Sequence` attached to its `Instantiated-Form` node has no component that matches the node, then it is not used to define `Form` nodes. The GCG must manage these components to help the graphical interactions with the `Instantiated-Form` they describe. The components shared by several `Instantiated-Form`

nodes which depend of the same `Sequence` are therefore within this `Sequence` or inside `Daughter-sequences` associated with this `Sequence` (see Fig. 5).

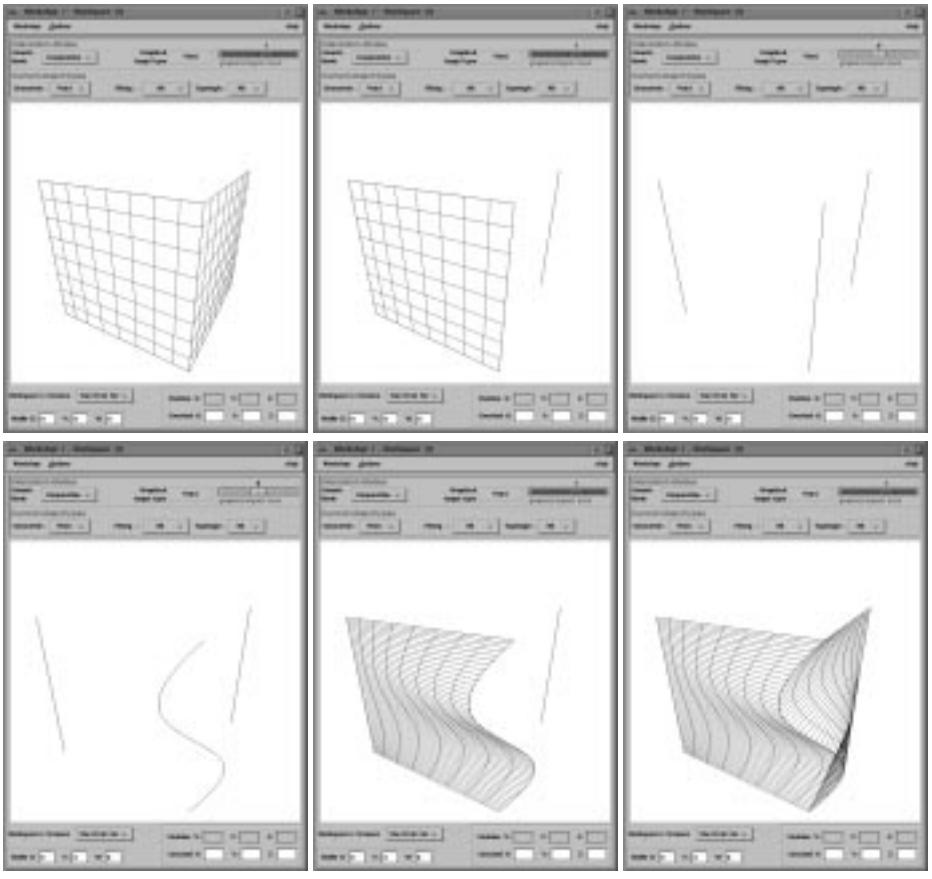

Fig. 6. Interactive steps required in a conventional modeler for transforming two `planar Surface` objects sharing a `segment` into two `free Surface` objects sharing a `free Curve` (left to right, top to bottom).

Additionally, each GCG is associated with one 'database project'. Following the idea that the GCG must help the object design activity, the graphical deletion or the moving of any `Instantiated-Form` never implies the destruction of this GCG node. In fact, these actions on MIX 3D objects are managed through state modifications of some GCG nodes. If an `Instantiated-Form` is not recovered after a given time delay, some data representations are transferred from the current GCG to its database. This approach allows the combinatory control of

the GCG, while giving the user the possibility of recovering any previous step of his work.

We also need to consider the role of the GCG with respect to the multimodal user interface. The GCG has an important contribution to apply the criteria of *logical complementary* and *data structure completeness* of the multimodal user interface manager (see section 2.3). It is the only place where the user's interactions on 3D virtual objects are decided to be valid or not. It manages the semantic aspects of the applications, but its work is 'amodal'. It can thus be considered as the 'melting-pot' of the multimodal user interaction.

3.3 Reactive Behaviour of MIX 3D Objects

All MIX 3D user interactions are defined with respect to several interactive modes. Some of them identify the *current object* type which is managed by the user interface, while a *generic mode* makes a distinction between three main contexts of interaction: *simulation*, *composition* and *instantiation* (see Fig. 10 and section 4.2). The *simulation* context is a direct interactive mode to study the evolution of the geometrical, fitting and topological properties of any 3D object, with real-time feedback. We present here a simple example of reactive behaviour that the GCG allows on MIX 3D objects within this simulation mode.

Suppose we have two `planar Surface` objects defined by three `segment` objects (first picture of Fig. 6). With a classical CAD system data structure, several interactions are required to modify the `segment` shared by the two `Surface` objects and to transform them into the two `free Surface` objects of the last picture of Fig. 6. For instance, it is necessary to delete the initial `planar Surface` objects and their shared `segment`, to build a `free Curve` from this `segment` and some input `Point` objects, to finally construct the two `free Surface` objects.

These transformations are made in several interactive steps (Fig. 6). In contrast, the reactive behavior of 3D objects in MIX 3D allows these transformations to be performed in one single step (Fig. 7).

The management of such behavior within the GCG is shown in Fig. 8. The shared `segment` ($c0$ node) being defined from a `Sequence` of two `Point` objects ($Seq0$ node), any input `Point` on this `Sequence` creates a new `Sequence` ($Seq0'$ node) to which 'dynamic inheritance' is applied until the `Instantiated-Form` level ($c0'$ node). According to the knowledge carried by the GCG nodes, semantic and data type conversions are managed for `Form` and `Instantiated-Form` nodes. For instance, since $c0$ is a `segment` (i.e. an `opened Curve`) and there is at least one input `Point` within the $Seq0'$ `Sequence`, the default semantic aspect of the resulting $c0'$ `Curve` is supposed to be `opened`, `free` and `cubic`. Additionally, the semantic specialisation of the `Curve` type allows $c0'$ to know that its default data representation is a `SplineCurve`. Recursive propagation of this management in the GCG is sufficient to obtain directly $S1'$ and $S2'$, the two `free Surface` objects sharing $c0'$.

More complex situations of reactive 3D objects suppose that the propagation which takes place within the GCG uses a 'width first' strategy and must respect

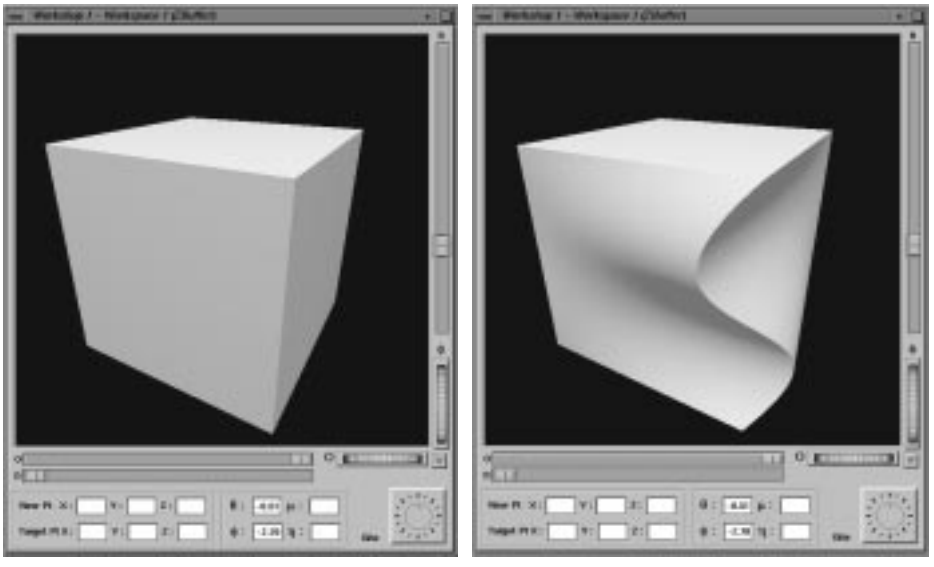

Fig. 7. Immediate transformation allowed by the GCG for the example of Fig.6.

in 'depth' the 'longest paths' of GCG compositions which depend of the current user interaction.

4 Cooperation Between Reactive 3D Objects and Multimodal Events

Using the concept of reactive objects, we study here the contribution of multimodality for interacting with a CAD system in the design process of 3D objects.

4.1 Vocal Inputs as Shortcuts for Menu Command Selection

One of the simplest ways of using multimodality is to exploit the equivalence of certain modalities, where either of two or more modalities can be used for the same interaction. We use this form of cooperation so that a user may issue commands which are normally menu-based by using a vocal equivalent. MIX 3D uses many commands which are laid in menus and cards (see Fig. 9). No matter how much care is put into conceiving these menus and cards, the user will, at least initially, waste time looking for the right command. Furthermore, this requires the user to shift his attention from the 3D working area to the menu and card boxes.

In order to limit the amount of attention shifting, we have provided the user with a set of vocal commands which can be substituted for menu and button

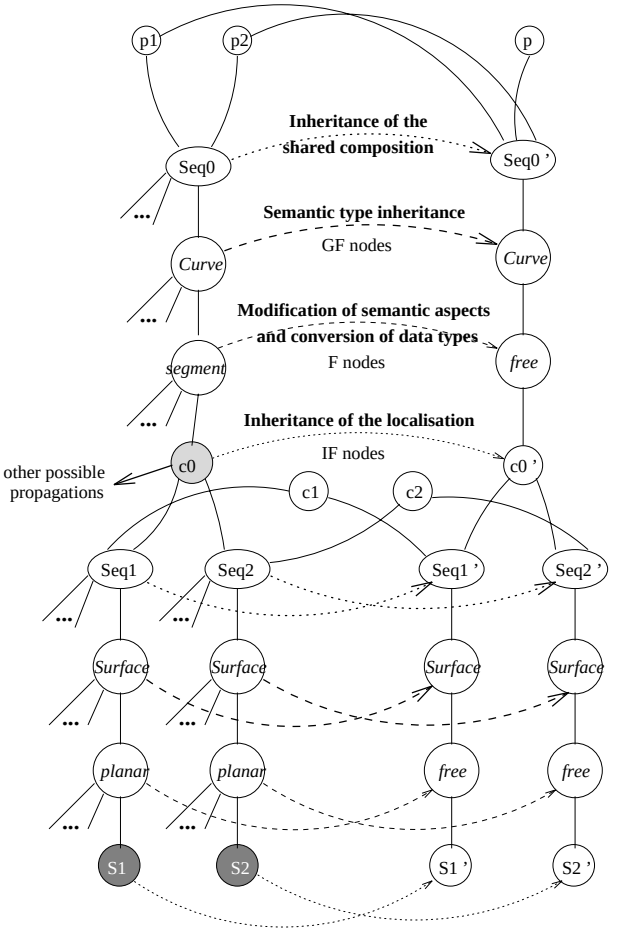

Fig. 8. Management of reactive behaviour in the GCG.

actions. Interaction can then take place entirely in the working area and the user can apply commands whose corresponding button or menu item are not actually visible. Of course, the user still has to learn the appropriate vocabulary but by using synonyms and abbreviations this can be made easier. The only problem then is the limitation in the size of vocabulary that current real-time voice recognition systems can handle. We estimate that the vocabulary for a full-fledged CAD system would contain over a thousand words, a lot more than our Datavox[1] system can handle.

[1] Datavox is a speech recognition system created at the LIMSI-CNRS laboratory and distributed by VECSYS.

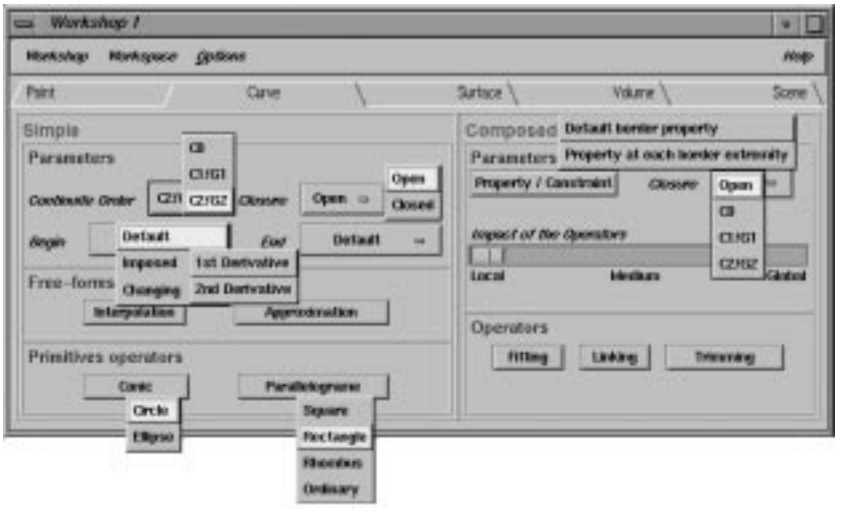

Fig. 9. Commands card and some of its sub-menus for the `Curve` type.

4.2 Just Name It: Vocal Modality for Deictic Interaction

The use of a variety of modalities can help the application to remove ambiguity from the user's actions. Figure 10 shows the control board of MIX 3D for managing the direct interaction of the user within the working area. In section 3.3, we already discussed the *simulation* context of interaction. But the *generic mode* menu gives the user two other contexts of interaction: the *composition* mode to manage the creation of new user classes (i.e. `Form` nodes) and the *instantiation* mode to build new MIX 3D objects (i.e. `Instantiated-Form` nodes) from any of these user classes. On the other hand, a deictic interaction is also defined according to the types of the *graphical target* and the *current object*. The former is automatically determined from the event selections and gives the interaction manager information about the precision of the object selections,[2] while the latter is fixed by the user to specify which geometrical, fitting or topological type of MIX 3D objects he wants at a given moment during the deictic interaction.[3] Furthermore, the difference between the *current object* and *graphical target* types defines the *graphical impact* level of a selection (in Fig. 10, the feedback at this level is the current value of the horizontal scale).

These menus and widgets are only here to help the user to remove the deictic ambiguities that the high level of knowledge modeling implies for MIX 3D objects. When the number of combinations of the possible ambiguities increases,

[2] At the present time, the event selections which find the closest `Point` or the closest `Curve` are the only two *graphical target* types implemented in MIX 3D.

[3] *Current object types have logical dependencies; for instance, the* `Edge` *and* `Border` *types are attached to the* `Curve` *level, but conversely a* `Curve` *(resp.* `Border`*) type does not imply a link with a* `Border` *(resp.* `Edge`*) type (see section 3.1).*

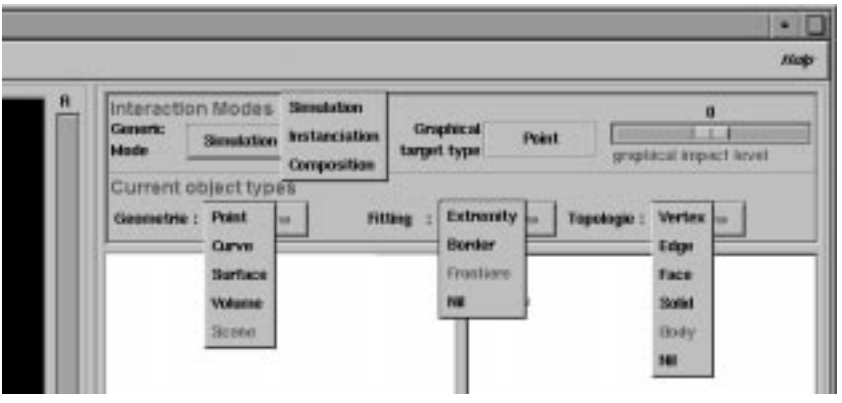

Fig. 10. Menus and widgets for controlling interactive modes.

the classical solution, which requires the human operator to use keyboard modifiers when selecting objects, becomes awkward for simple ergonomical reasons. In fact, the only powerful alternative is the use of the vocal modality. For instance, by making deictic references the user would be able to say *"check this border"* when clicking near a `Curve` which describes a `Border` of a `Surface`, and the application would activate a set of interactive cards according to the fitting properties of this `Border`. Taking this approach even further, the user would be able to give names to objects and refer to them later using these names. This should prove useful for managing complex virtual scenes.

As we saw before, the GCG is already able of managing the reactive behaviour of MIX 3D objects. The use of the vocal modality makes it possible to avoid latencies in interacting with reactive virtual objects. In other words, the vocal input associated with reactive objects is convenient to support 3D simulation activities in object design.

4.3 Just Draw It: Sketch Recognition as a Non-Standard Modality

Another way of using multimodality is to combine modalities to perform a complex task. Within MIX 3D, we are introducing a multimodal sketching interface called PADEM, which tries to simulate the familiar working context of a draftsman (paper and pen) with a tactile screen and an electronic pen. But as hand drawn sketches may have several semantic interpretations, we choose to assist the sketch recognition process by means of vocal interaction. This research is presently focused on 2D drawing recognition.

The principle of this sketch recognition lies in two sub-processes. The first one manages a 'signal analysis' and a recognition of 'single' sketches (Fig. 11). By 'signal analysis' we mean that the set of pixels of a 'single' sketch is segmented according to geometrical features (C^0 continuity, alignment of pixels, curvature, and so on) from which semantic aspects of the `Curve` geometrical type are identified as well as fitting properties (see section 3.1). The second process makes a

'contextual analysis' of the resulting objects to improve recognition scores. Some ergonomical strategies are also used; for instance, the `polygon` recognition cases of Fig. 11 take in account the fact that the first pixels of a sketch are more important than the others, because the precision of a drawing gesture is generally better at its beginning. Additionally, the fitting properties are also used to perform the drawing recognition within both sub-processes; for instance, specific graphs of `CurveFitting` objects identify the semantic aspects of any `polygon`.

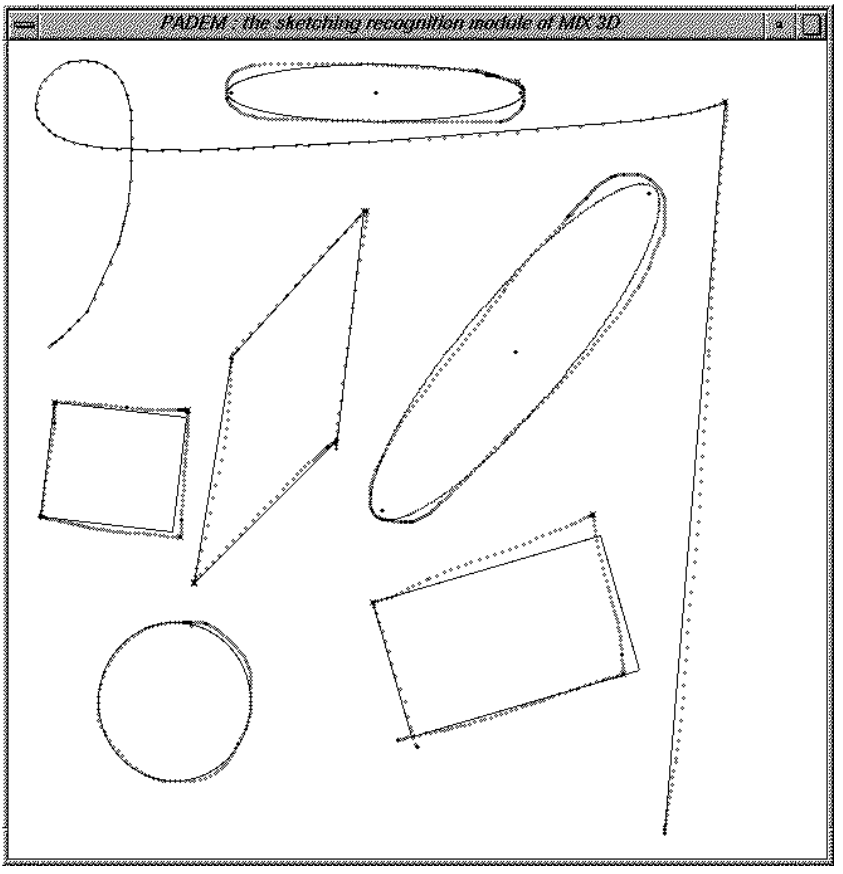

Fig. 11. Examples of 2D drawing recognition for 'single' sketches.

In many cases, several semantic interpretations are possible, simply because nothing is more different than two similar hand drawn sketches! When the drawing recognition process fails, PADEM will allow the user to correct this with vocal inputs, specifying the semantic aspects the shape must have. This extra modality will give the human operator the opportunity of supervising the sketch recognition process.

4.4 About Vocal and Textual Output

Just as multimodality takes input interaction a step further, it can also enhance output interaction to provide the user with more accurate information, by making use of the properties of the various modalities and by combining some of them to produce output messages (Krus, 1995).

For example, a message can be presented using text and vocal output. This presentation may have redundancy and/or complementarity effects. Text can be used to confirm information in the vocal message, while voice allows the user to remain focused on his job (which is very useful for simulation activities). Indeed, these modalities have different properties which influence the way they are perceived by the operator. Textual output is persistent and may contain detailed information which can be accessed at a later time. Vocal output, by contrast, is short-lived and can convey less information than text, but it will get the user's attention more easily, especially if he is already busy looking at some part of his work. Thus, use of the characteristics of the modalities and of cooperation between these modalities can produce efficient presentations.

At this time, MIX 3D uses these considerations in a simple way: feedback messages from most user commands are in textual form. For some commands which do not produce visual results a prerecorded vocal message is also sent. We are currently considering the value of adding more vocal outputs, in particular as an option for menu commands. Carefully chosen messages could in effect help the user learn the correct vocabulary for the vocal commands that he can use for input. This kind of loopback should prove very valuable.

5 A Real-Time Multimodal User Interface Architecture

In order to implement the kind of interaction required by multimodal applications, we have developed a software architecture which was designed to be efficient, portable and extendable. This is a distributed architecture where use of load-sharing ensures near-real-time performance. Figure 12 describes this architecture, which is based on the X Window library and the widget toolkit (Nye, 1989; Nye and O'Reilly, 1989). It extends these low-level components with new modalities and accurate dating and ordering of events, so that high-level multimodal fusion modules can be implemented (Bellik et al., 1995b; Martin et al. 1995).

The architecture is divided into two parts. The *modality server* is responsible, along with the standard X server, for the dating and the delivering of events to the application. The *modality toolkit* is used by the applications to add multimodal event to widgets and can filter events based on their type. The toolkit guarantees that the handlers will receive events in the order in which they are produced.

For the remainder of this chapter, we will refer to modalities other than those provided by X Window (such as mouse and keyboard) as *non-standard* modalities and to the events they produce as *non-standard* events. We will first

describe the nature of non-standard modalities, then present both parts of the architecture.

5.1 Non-Standard Modalities and Events

Non-standard modalities usually require a recognition process before events can be produced. For example, voice events (words) need to be recognised from the speech signal. Such recognition processes are typically heavy tasks which require a lot of machine power. As such, it seems unfeasible for them to reside in a system which requires near-real-time response for other tasks such as managing the interaction between user and application. We have therefore decided to implement these processes on slave machines which only send the results of their recognition to the modality server. Communication between slave processes and the server can be done via a serial link or a network connection. Recognition results are sent to the modality server along with additional status information.

The events produced by non-standard modalities have the interesting feature of having a length in time; events like words have distinct beginning and ending dates. We treat standard events as having a zero length in time, i.e. their beginning and ending dates are the same.

Our architecture is currently validated for one non-standard modality: the Datavox voice recognition system, installed on a PC. It transmits the words it has recognised, along with a score and a time frame number for the beginning and the ending of each word. Additionally, we support a tactile screen by using the *X Window Input Extension protocol* (Nye, 1989) for sketch recognition.

5.2 The Modality Server

The modality server is responsible for dating the events it receives from the recognition modules. It sends these events to the application that currently has the input focus (as defined by the X Window system) using the X server.

The modality server is organised in modules. As described in Fig. 12, a *modality module* is defined for each modality. These modules have the appropriate know-how to compute the date for the events. The entire process of computing the date and sending the events takes several steps.

First, the transport module receives events from the serial or network ports, and sends them to the appropriate modules based on a type identifier attached to the messages. This module then computes the beginning and ending dates for each event. This process is modality dependent (and even recognition system dependent). The module should take into account the time that it took for the recognition system to recognise the events and for these to travel to the server. A recognition system can produce several events for one signal (like several words in one sentence). These events are sent together to the modality server in one message, and the modality modules are responsible for producing individual events out of that message.

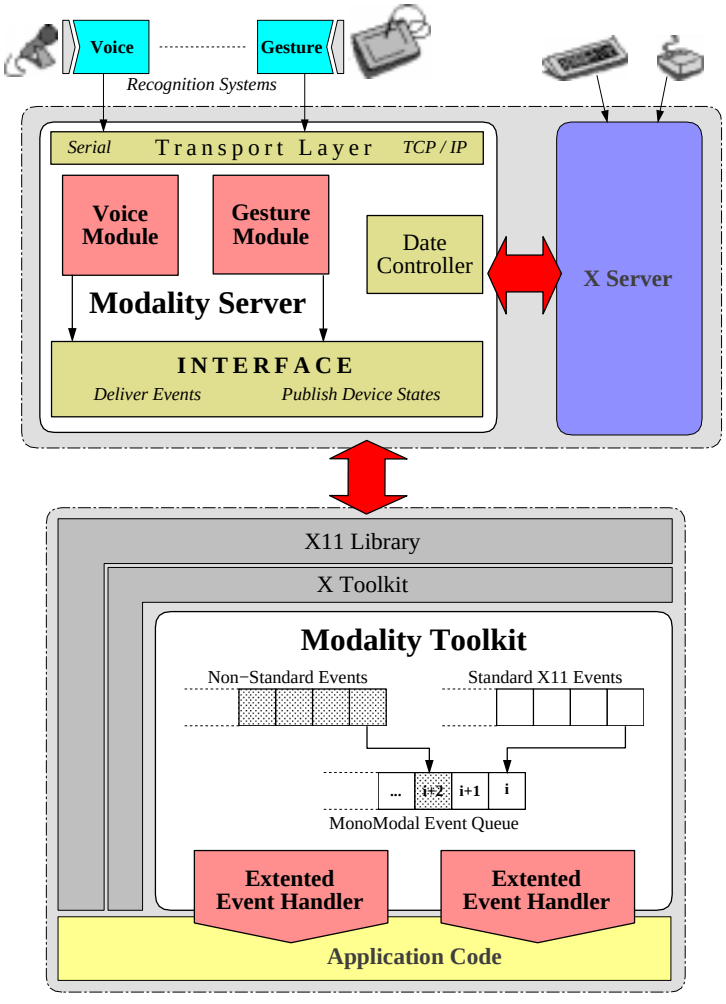

Fig. 12. Architecture for 'Real-Time' Multimodal User Interface based on X Window.

An important part of the server is the *dating module*. This a responsible for maintaining an accurate equivalence between the Unix idea of date (which is used to know when an event arrived to the transport module) and the X Window idea of date (which we use for dating the events, standard or non-standard). Since these two representations are not the same, the dating module constructs an equivalence at start up, adjusts it continuously while it is running, and can make conversions either way at any time. Once the events are dated, they are passed to the *interface module*. This module uses the X server to send them to the appropriate application.

As we saw in the previous section, recognition systems can also send messages indicating what state they are in. These messages are treated in much the same way. The modality modules receive them, but they then instruct the interface module to *publish* that information for clients to read it at any time. This publishing is done using X properties for the modality server's window (Nye, 1989).

5.3 The Modality Toolkit

This toolkit extends the standard X toolkit (`Xt`) to support an extended event handler. This handler can request events (standard or non-standard) based on their type and is guaranteed to receive them in the order that the signal for the events was produced, *not* the order in which the signal was processed. This is crucial to ensure an accurate fusion of monomodal events.

The toolkit contains a main controller which keeps track of which widget currently has the input focus and dispatches the events. The dispatching differs from the normal `Xt` mechanism only for non-standard events.

The toolkit also builds a controller for each widget that requires an extended event handler. Each controller contains an event queue where standard and non-standard events are stored as they arrive. They are ordered in this queue based on their beginning date.

Since events produced by the keyboard or the mouse are produced very rapidly by the hardware, they arrive sooner than voice events even though the signal for these might have been produced earlier. The controller for each widget ensures that no event is delivered while a non-standard modality is analysing a signal to produce new events. Every time the controller receives an event, it stores it at the appropriate position in the queue. It then scans all devices, using the modality server, to know if any of them are currently analysing a signal.[4] If none of them is building a new event, then the event at the head of queue is popped and delivered to the extended event handlers that have requested that type of events. On the other hand, if an event is being produced by a modality, then the controller holds the events in the queue until user-specified delay expires.[5]

5.4 Other Requirements for a Multimodal User Interface System

The architecture we have developed can be used as a basis for a complete multimodal user interface system. It contains most features required by multimodal fusion systems, for which various architectures have been proposed (Bellik et al. 1995b; Martin et al, 1995). There is one additional issue that needs to be addressed by these systems. Applications typically have several widgets, even sometimes several windows (such as in our case). While the modality toolkit can

[4] Standard devices are never considered to be producing an event, since they do this in one single unit of time.

[5] Setting this delay to zero makes the controller deliver events in the order they arrive.

handle event queues for each widget so that chronological sorting and fusion can take place, what happens if interactions require events from different widgets to be merged? A simple example would be the copying of a 3D object from one window to another, using the *"put this here"* vocal command and two designations to identify the object and the new destination. Such high-level interaction require events to be propagated to the parents in the hierarchy of widgets. However, in most cases, this need not be done for all events and through all widgets in the path to the root widget. But identifying the appropriate events and widgets seems too application-dependent to be integrated within our architecture, as it requires contextual knowledge of the interaction and the application. Thus, multimodal fusion systems and/or application developers should be aware of this when designing the interface.

As mentioned before, our architecture needs to be extended in several ways. First, it should probably also manage the input signals for non-standard modalities, delivering them to the appropriate slave processes. For example, the signal from the microphone could delivered to the recognition process. This would be the first step toward a multimodal terminal.

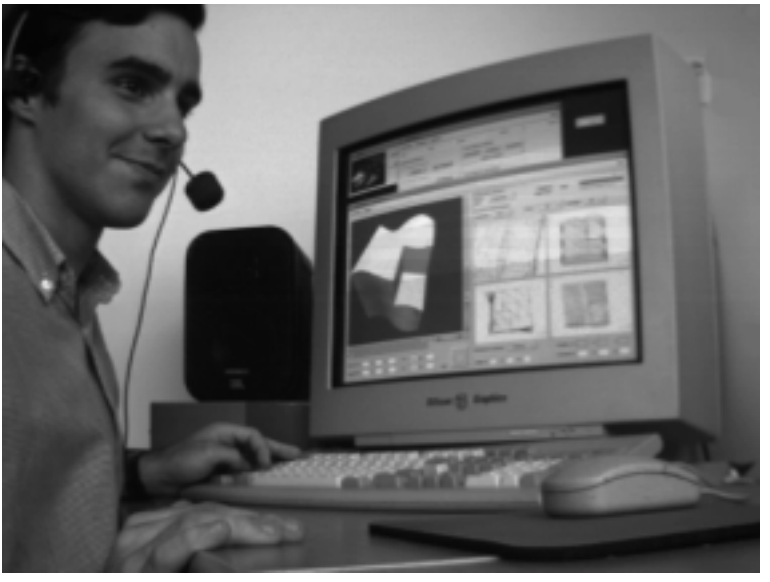

Fig. 13. MIX 3D's multimodal environment with the input and output voice devices.

But the most important extension would be to support multimodal output. As specified in (Krus, 1995), multimodal presentations are more efficient for displaying complex or critical information. Furthermore, they should be adapted to take into account the characteristics of the message, of the task that is being

performed, of the user, and of the environment where the interaction is taking place. Such multimodal presentations systems also have requirements for the selection, the combination, the layout and the synchronisation of modalities which should be integrated in our system.

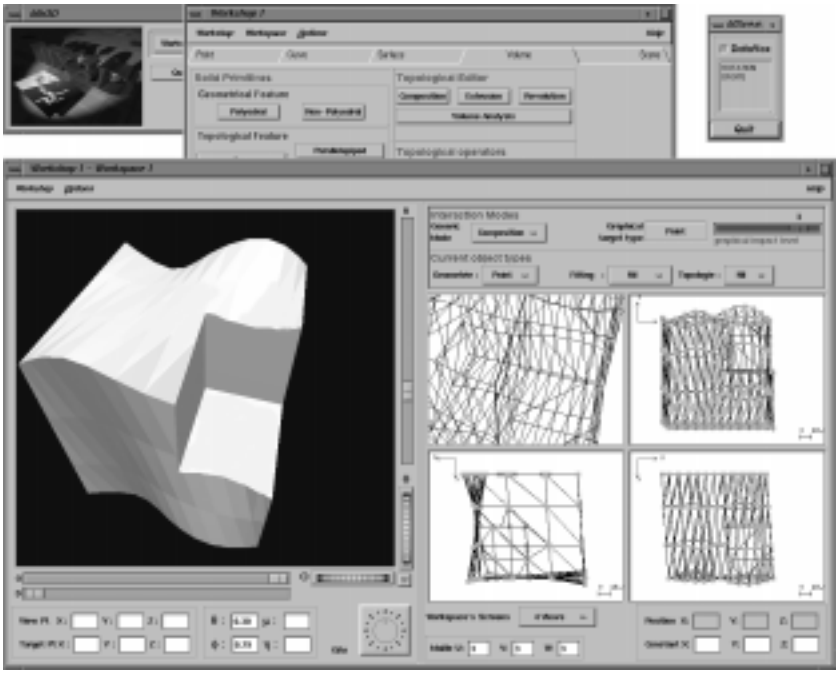

Fig. 14. X Window user interface of MIX 3D and **Modality Server** for vocal input recognition (top-right corner), with a `Solid Volume` described by `free Surface` objects and resulting of a topological cut operation between two previous `Solid Volume` objects.

6 Conclusion

We have analysed and presented the requirements for a next generation of CAD software, an advanced 3D system which allows graphical and spatial simulations for virtual objects. Using powerful multimodal interfaces, MIX 3D, our working prototype (see Fig. 13), gives life to objects with behaviour. Based on the characteristics of these objects and the tasks that designers perform, simulations and manipulations can now easily be applied to objects while they are being designed. We have shown that these objects must have fast reactive behaviour and an appropriate knowledge representation in order to allow the user to perform these

simulations in real time with multimodal interaction combining graphical actions and vocal commands. To manage object simulations, we have elaborated a powerful 3D object representation model combining geometrical and topological knowledge modeling with the Generalised Constructive Graph (a generalisation of CSG trees). In order to support high level interaction, we have created a distributed architecture based on the X Window system. It provides most services required by multimodal fusion systems.

As Fig. 14 shows, the working MIX 3D prototype is now turning into a full-fledged CAD package which can handle solid modeling with objects described by free-form surfaces. Consequently, a short-term objective is to perform ergonomical experiments in order to evaluate the contribution of cooperative tools, reactive 3D virtual objects, and multimodal user interfaces in a real working context. In parallel, more advanced interaction techniques are being introduced. Sketch recognition is improved, and 3D gesture recognition (using a numerical data glove) is considered. A complete multimodal fusion system will integrate all these modalities.

Acknowledgements We wish to thank L. Arnal, J.P. Di Lelle, and F. Ledain for their work on MIX 3D and their help in producing the images for this chapter.

References

[1999]Barr, A. H. (1984) Global and local deformations of solid primitives. *Computer Graphics*, 18(3), 21–30.

[1999]Baumgardt, B. (1972) Winged-edge polyhedron representation. Technical Report CS 320, Dept. of Computer Science, Stanford University.

[1999]Bellik, Y. and Teil, D. (1993) A Multimodal Dialogue Controller for Multimodal User Interface Management Systems. Application: a Multimodal Window Manager. In *Proc. INTERCHI'93*, New York: ACM Press, 24–29.

[1999]Bellik, Y., Ferrari, S., Néel, F. and Teil, D. (1995) Requirements for multimodal dialogue including vocal interaction. In *ESCA Tutorial and Research Workshop on Spoken Dialogue Systems*, Hanstholm (Denmark), May 1995, 161–164.

[1999]Bellik, Y., Ferrari, S., Néel, F., Teil, D., Pierre, E. and Tachoires, V. (1995) Interaction Multimodale : Concepts et Architecture. In *4èmes Journées Internationales sur l'Interface des Mondes Réels et Virtuels*, Montpellier (France), June 1995, 37–45.

[1999]Carlson, W.E. (1982) An algorithm and data structure for 3d object synthesis using surface patch intersections. *Computer Graphics*, 16(3), 255–263.

[1999]Casale, M.S. (1987) Free-form solid modelling with trimmed surface patches. *IEEE Computer Graphics and Applications*, January 1987, 33–43.

[1999]Coutaz, J. et al. (1992) Interfaces multimodales et architecture logicielle. In *Workshop Report of IHM'92, 4èmes Journées sur l'Ingénierie des Interfaces Homme - Machine*, Paris (France), December 1992, 9–44.

[1999]Gaildrat, V., Vigouroux, N., Caubet, R. and Pérennou, G. (1993) Conception d'une interface multimodale pour un modeleur déclaratif de scènes tridimensionnelles pour la synthèse d'images. In *2èmes Journées Internationales sur l'Interface des Mondes Réels et Virtuels*, Montpellier (France), March 1993, 415–424.

[1999]Krus, M. (1995) Présentation multimodale d'information. Master's thesis, Université Paris XI, Orsay (France).

[1999]Lebahar, J. C. (1992) Quelques formes de planification de l'activité de conception en design industriel. *Le Travail Humain, Presse Universitaire de France*, 55(4), 329–351.

[1999]Levin, J.Z. (1980) Quadril: a computer language for the description of quadric surface bodies. *Computer Graphics*, 14(3), 86–92.

[1999]Macé, P. and Bourdot, P. (1991) A method to control continuity: Application to easy patches fitting. In *International EUROGRAPHICS Workshop on Computer Graphics and Mathematics*, Genova (Italy), October 1991, 150–169.

[1999]Martin, G. L. (1989) The utility of speech input in user-computer interfaces. *International Journal of Man-Machine Studies*, 30, 355–375.

[1999]Martin, J. C. Veldman, R. and Béroule, D. (1995) Towards Adequate Representation Technologies for Multimodal Interfaces. In *CMC/95, International Conference on Cooperative Multimodal Communication*, Eindhoven (The Netherlands), May 1995, pages 207–224. *Reprinted in revised form as Martin et al. (1997) in this volume.*

[1999]Martin, J. C. Veldman, R. and Béroule, D. (1997) Towards Multimodal Interfaces: a Theoretical Framework and Guided Propagation Networks. *This volume.*

[1999]Nigay, L. and Coutaz, J. (1993) A design space for multimodal systems: Concurrent processing and data fusion. In *Proc. INTERCHI'93*, New York: ACM Press, 172–178.

[1999]Nye, A. (1989) *Xlib Programming Manual (Vol. One of The X Window System Series)*. Sebastopol, CA: O'Reilly and Associates.

[1999]Nye, A. and O'Reilly, T. (1990) *X Toolkit Intrinsics Programming Manual (Vol. Four of The X Window System Series)*. Sebastopol, CA: O'Reilly and Associates.

[1999]DeRose, T. D. and Barsky, B. A. (1985) *An intuitive approach to geometric continuity for parametric curves and surfaces.* In *Computer-Generated Images - The state of the art*, edited by N. Magnenat-Thalmann and D.Thalmann, Heidelberg: Springer, 159–175.

[1999]Shah, J. J. (1990) Philosophical development of form feature concept. In *Proc. of the Features Symposium, Computer Aided Manufacturing - International, CAM-I 90*, Boston (USA), August 1990, 113–128

[1999]Schomaker, L. et al. (1995) A taxonomy of multimodal interaction in the human information processing system. Technical report, ESPRIT PROJECT 8579 MIAMI, February 1995.

[1999]Tilove, R. B. and Requicha, A. A. (1980) Closure of boolean operations on geometric entities. *Computer Aided Design*, 12(5), September 1980, 219–220.