Interaction Facilities and High-Level-Support for the Exploration of 3D-Models: System Architecture

Bernhard Preim

Institut für Simulation und Graphik, Otto-von-Guericke-Universität
Magdeburg, Universitätsplatz 2, D-39106 Magdeburg
e-mail: {bernhard}@isg.cs.uni-magdeburg.de

Abstract. We describe the design and architecture of an interactive system to illustrate spatial phenomena. Our system, the ZOOM ILLUSTRATOR, focuses on the combination of rendered images with different levels of textual descriptions for educational purposes. Fisheye Techniques are employed to present detailed information while maintaining the global context of the overall available information.

The design of our system was guided by the object-oriented approach. To encourage comparison, it is furthermore described in terms of the reference model for intelligent multimedia presentations. The feasibility of this approach encourages the use of the reference model for the design of interactive systems. Our system targets at interactive manipulation by an end-user, however, it offers automatic support to enhance its usability. Automatic support is needed for the coordination between images and text, for the annotation of a 3D-model and for the synthesis of a layout.

Keywords: Interactive Illustrations, Image-Text-Relation, Fisheye-Zoom-Techniques, Multi-modal Presentation, Object-Oriented Design

1 INTRODUCTION

Interactive 3D-graphics bears a high potential for the explanation of complex 3D-phenomena, as can be found for example in engineering and anatomy. Modern hardware allows rendering 3D-models with considerable detail at interactive rates. The interactive handling of 3D-models is important to clarify spatial relations. While this is well-recognized, especially in the computer graphics community, not enough effort has been spent on the combination of rendered images and textual information.

Borrowing from textbooks gives hints on how to combine images and text. Images often are surrounded by labels referring to their parts via reference lines. Explanations refer to the spatial structure and are enhanced by cross references as to spatial relations. In textbooks, however, explanations are generally not integrated in an illustration but are placed under an image or even on a separate page, which complicates comprehension.

Interactive systems can handle this problem and tailor the presentation to the information requested. In current hypermedia systems, however, this often results in the display of multiple windows, the management of which imposes a high burden on the user. To make matters worse, an obvious coordination of explanations and images is missing.

Based on these observations we developed the ZOOM ILLUSTRATOR. Our system tackles some problems which are typical in multimodal presentation systems, as there are:

- The use of different modalities, namely images and text, must be coordinated. In particular, an image and its textual description must fit conceptually.
- Flexible layout strategies are required. This is due to the amount of textual information related to a complex 3D-model which cannot be displayed at once.
- User support is necessary for the annotation of graphical objects with textual labels.

To adapt a presentation to the user's interest, Fisheye Techniques, as introduced by Furnas (1986), are very useful. Fisheye Techniques place and scale the space to accommodate information depending on the user's interest and allow to look at something in detail while maintaining the context and thus the overview.

Important information is emphasized, driven by a *degree-of-interest* (DOI) which is assigned to each piece of information. DOI-values depend on a static à priori importance (API) and on dynamic factors which consider the distance of pieces of information from the one the user interacts with. This distance, however, is not necessarily a spatial distance in terms of screen coordinates.

This paper describes the design and the architecture of an interactive system with focus on the coordination of graphics and textual descriptions. Fisheye Views are generated to integrate detail and context. According to its purpose, to integrate rendered images with textual descriptions, that is to *illustrate*, and its basic interaction technique, to *zoom*, it is called ZOOM ILLUSTRATOR.

In this paper we put emphasis on the Object Oriented Design (OOD) and the overall architecture of the system. Because the OOD as well as our architecture use a dedicated terminology, a system description in terms of the reference model for IMMPs (see Rugieri *et al.* (1996)) is presented.

Our system targets at interaction facilities for an end-user. However, we believe that there is not a strict separation between knowledge-based systems and interactive systems, because the latter can benefit from automatic support to allow high-level interaction. It turns out that the reference model is able to capture our design and offers new ways to look at the system.

2 RELATED WORK

Our work is related to systems which consider communicative intents to generate 3D-illustrations. Planning strategies are used for the application of illustration rules and the selection of modalities. Advanced techniques for the automated design of 3D-illustrations were developed in the IBIS-System and in the WIP-Project.

The IBIS-system (*Intent Based Illustration System*), described in Seligmann and Feiner (1991), is based on an extensive study of the work of technical illustrators. IBIS employs 3D-models with considerable detail. Transparency and cut-aways are used as well as insets (small details scaled up in a large image). Arrows emphasize parts which are important with respect to the goals specified. We learned from their work that visibility and recognizability of important objects are crucial for illustrating 3D-models. With IBIS excellent images can be produced. However, labels or other textual descriptions are not included.

The WIP-project (*Knowledge Based Information Presentation*) is described e.g. in Wahlster *et al.* (1993). It originally targeted at static illustrations, which requires sophisticated strategies to select the content and modalities to communicate the intents specified. In a later stage of the project, Rist *et al.* (1994) describe a system for the semi-automated illustration design. The system offers interaction facilities to incrementally augment and evaluate an illustration the system suggested. The final illustration generated, however, is not intended for interactive usage. André and Rist (1994) extend the project to combine interaction facilities and knowledge-based techniques for the handling of illustrations.

IBIS and WIP offer strategies to plan, evaluate and replan an illustration. From the intents specified, constraints are derived and solved. The overall goal is to produce final illustrations which fulfil the communicative intents.

Our system differs from previous work as the focus lies on providing flexibility to interactively explore an illustration and does not target at a final illustration. The need for planing strategies is reduced because the user can interactively build up a presentation with a specific view on the 3D-model and the text displayed. Instead, the user-interface gets more important. While the strategies to generate an initial layout are straightforward, sophisticated techniques are developed to place textual items after interaction. Finally, a difference lies in the application area: Our system focuses on complex, irregular shapes, as can be found in anatomy.

Despite these differences some similarities exist: To emphasize an object graphically which is textually explained, illustaration techniques as used in WIP and IBIS are helpful. This includes the selection of a perspective and the use of semi-transparency to show an object behind another. Furthermore transparency is used to de-accentuate objects which are not important in the current context.

Besides intent-based illustration systems, applications of Fisheye-Techniques are related to our work. We are inspired by Noik (1993), who applied Zoom Techniques within the hypertext domain. Especially our DOI-calculation is related to Noik's work, which employs a conceptual distance between nodes, derived from the existence of hyperlinks. Dill *et al.* (1994) developed the continuous zoom which provides smooth transitions between an original layout and the layout after a zoom step. Continuous transitions indeed enhance the user's understanding and encouraged us to use a variant of this algorithm.

3 ARCHITECTURE OF THE ZOOM ILLUSTRATOR

With our design, we combine interactive 3D-graphics with hypertext functionality for educational purposes. This involves the presentation of more or less detailed information on the textual part and results in a continuous zoom of the corresponding node to accommodate this information, while automatically repositioning and rescaling other nodes.

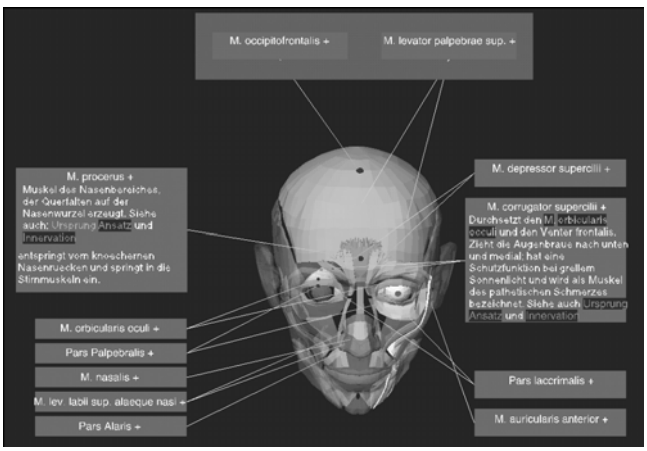

Figure 1: Example layout of the ZOOM ILLUSTRATOR with different levels of text presentation ranging from a mere rectangle (on the right) to an extended explanation on the right side. The material properties on the graphical part are adapted to the amount of text presented. The endpoints of reference lines are marked with small spheres.

The annotation of the graphics is based on the structure of the 3D-model. To ensure a high quality, we employ 3D-models which are commercially available. This has one serious disadvantage concerning the annotation: we must live with the vendor's structure of the model, which tends to be very coarse and unsuitable to assign textual information. Therefore the structure must be tailored before the reference points can be calculated. Each graphical object is connected to the corresponding part in the textual information via a line.

Navigation through textual information is supported by hyperlinks. These hyperlinks connect different explanations for one node as well as explanations of different nodes with each other. As usual, e.g. in Web-Browsers, the colour of hyperlinks is adapted to whether or not they have been visited.

On the graphical part, 3D-models can be transformed freely. Furthermore, changes on the textual presentation cause an adaptation of the corresponding graphical part. Transparency and saturation of colours have proven to be important parameters to adapt an object's appearance to the amount of detail of its textual description. Figure 1 shows a typical output of the system which incorporates this adaptation.

For the layout, different strategies are necessary for the initial layout and the layout after user-interaction. The initial layout encompasses the labels with the highest API-values together with one instance of the 3D-model to explain. The API-values (recall Furnas (1986)) are derived from the object structure (the position within the hierarchy) and from geometric criteria (size and visibility of objects). The *Layout Manager* is responsible for choosing the most important nodes to label. (see the architecture in Figure 3). Textual information is distributed as evenly as possible between the left and the right side of the image.

Originally, the zoom algorithm placed and scaled the whole space to accommodate textual information. In many cases, the zoom performs well and does what can be expected. However, if one node is zoomed up, others may be closed, which is an unwanted sideeffect. To prevent nodes from being closed due to a zoom step, they can be transferred to the *pinnwall* (see Figure 2). This is a container for nodes which are always connected to the graphics. These are at fixed positions and thereby somehow "privileged" because they are not exposed to the zoom. However, the nodes managed by the *pinnwall* be enlarged to display explanations.

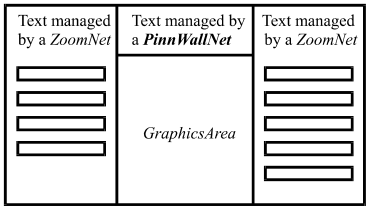

Figure 2: Layout for the illustration of one 3D-Model

Figure 3 summarizes the architecture of the ZOOM ILLUSTRATOR. The generation of an illustration is based on two sources (see the top vertical boxes in Figure 3, next page):

The first is a *Scene Description*, containing a polygonal 3D-model which is structured into objects. Secondly, we employ a file with related *Textual Descriptions* referring to the objects in the scene description. The *Textual Descriptions* contain labels and explanations.

After the sources are loaded, an internal representation is generated using the correspondence between a common key in the *Scene Description* and in the *Textual Description*. A user can interact with the *Text Display* (ask for explanation, follow hyperlinks) and with the *Image Display* (transform the model, adapt individual objects).

The architecture is well-suited to communicate the ideas behind the ZOOM ILLUSTRATOR. It is abstract insofar as it does not lead to an implementation in a straightforward manner. The terminology is dedicated to the basic features and techniques of our specific system. To encourage comparison an architecture in terms of IMMP-systems is presented in Section 5.

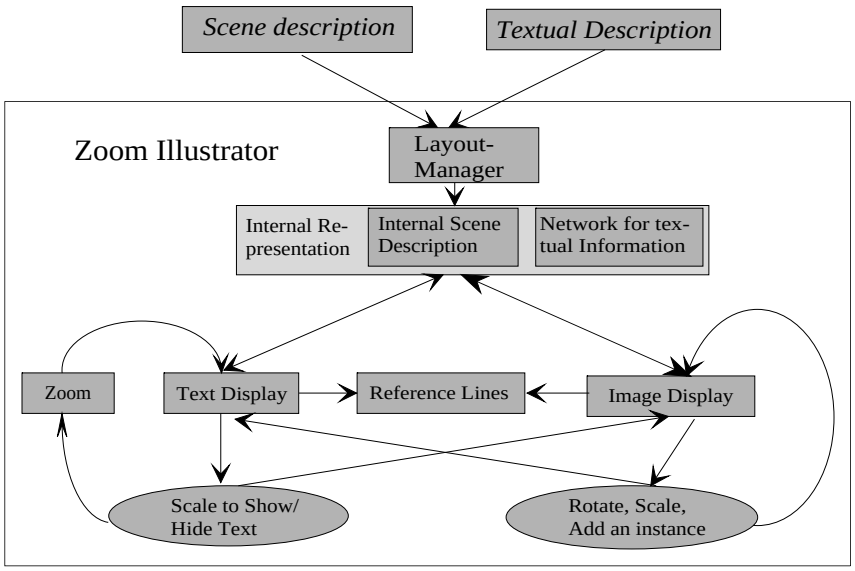

Figure 3: Overall Architecture

4 OBJECT-ORIENTED DESIGN

Following the conceptual architecture, this Section describes the Object Oriented Design of the ZOOM ILLUSTRATOR. *Classnames*, *methods* and *members* are written in italic. The spelling *Class::method()* refers to a method of a class. The relations between the most important classes are described in Figure 4.

Design of the toplevel-classes

The basic idea of the OOD is to regard an illustration as a hierarchy of parts which manage a media or a screen space. This implies that one instance is on top of the hierarchical composition of the illustration. To realize the described architecture, images and text must be managed. This gives rise to the classes *TextArea* and *GraphicsArea*. These classes manage the extent of their media and contain children, instances of a rendered image and networks of textual information. They assign space to their children, cause a rearrangement of their children (e.g. if a new child is added. Both classes involve some file operations, namely they load and process information from an external file).

The work of the *TextArea* and the *GraphicsArea* is coordinated by class *Illustrator*, which is on top of the hierarchical representation. It communicates directly with the application and distributes commands to their children (namely, *GraphicsArea* and *TextArea*).

The similarities between these three classes give rise to define a common base class which we call *IllustratorPart*. It has the members *children* and *screenspace* and methods to manipulate them.

The peculiarities of the part which is on top of the illustration are summarized in a class *TopLevelPart* which is derived from *IllustratorPart* and enhances the inherited behavior by methods to manipulate the viewing window in which the whole illustration is presented as well as global rendering parameters.

With this policy a strict separation of concerns is achieved. The basic abstract class *IllustratorPart* which is very generic provides the basic services for all parts of an illustration, where some methods are pure virtual, requiring reimplementation in derived classes. Class *TopLevelPart* provides the generic behavior of that part which is on top of the illustration. Class *Illustrator* which is derived from Class *TopLevelPart* is less generic and concentrates on the coordination of images and text. Furthermore, Class *Illustrator* is responsible for the overall layout of the illustration. An even more dedicated class is derived from class *Illustrator* and provides the specific behaviour required to control the Fisheye Zoom. This derived class is called *ZoomIllustrator*.

Graphical Interaction

The *GraphicsArea* manages *3D-Models*. Each *3D-Model* instance has its own *transformation* to be rotated and scaled independently. Figure 5 demonstrates the independent handling of two *3D-Models*. *3D-Models* are containers for *3D-Objects*, with each object having its own material information. With this structure the material of each *3D-Object* can be changed independently in several instances of the *3D-Model*. Thus, an object can be emphasized selectively in one instance of the *3D-Model*.

Media Coordination

One important issue in a Multimodal Presentation System is the coordination of the different media. In the ZOOM ILLUSTRATOR, this task is carried out in the class *Illustrator*, which distributes commands at the highest level. Coordination at a lower level (between one piece of text and the related graphics part is carried out by a collaboration between *IllustratorNode* and class *3D-Object*.

Requests to change the view, to add or remove an instance of the *3D-Model* are propagated to the *TextArea* which initiates an update of all textual information and of all reference lines. The *TextArea* manages several *IllustratorNets* which "know" their extent on the screen and hold a list of *IllustratorNodes*. The *IllustratorNets* initiate the update process for their *IllustratorNodes*, which includes the adaptation of the presentation to the space available. Requests to explain a node are processed from the textual part, resulting in a zoom step to accommodate the desired text and from the graphics part. On the graphical part the *3D-Object*-instance to be explained is emphasized and those *3D-Objects* which may occlude it are deaccentuated performing changes of material properties. If necessary, the whole 3D-model is transformed to ensure visibility of the explained object.

Navigation in Textual Information

The *LinkManager* maintains all hypertext links currently presented an explanation. This allows to adapt the presentation of nodes (e.g. the label colour) if they are mentioned somewhere else in an explanation. A link is a relation between two nodes under a certain aspect, that is *node A* has an *aspect* (an explanation) which refers to *node B*. As shown in Figure 4, *IllustratorNet* is an abstract class. Subclasses have to define a layout strategy. Two subclasses are defined: *ZoomNet*, which uses the zoom algorithm for the layout and *PinWallNet*, with nodes at fixed positions. Instances of

ZoomNet manage the labels on the left and right side whereas an instance of *PinnWallNet* is responsible for the nodes on top of the image. The *Zoom* realizes the zoom algorithm, that is it manages intervals and initiates an update of all *ZoomNodes* according to a zoom step.

Controlling Text Layout

Important methods of class *TextArea* are *move()* and *rearrange()*. The *move()*-method transfers a node to another network by an animation. It is invoked for instance to shift a node to the *PinWallNet*. The movement consists of three independent and incremental changes: A zoom step to provide the space in the target network, a second zoom step to distribute the space which is no longer needed in the source network and the actual movement from the source network to the target area. The *rearrange()*-method redistributes textual information to prevent crossing reference lines (which may occur when transforming the 3D-model). This process is only invoked after an explicit command of the user to avoid confusions resulting from "flying" labels.

Managing Information for one node

An *IllustratorNode* manages application-specific data related to a node. This includes information about its appearance (e.g. colours of the label, of the rectangular area) and about its relation to the graphics part, summarized in an instance of *GraphicalReference*. This instance holds the reference object within the graphic (see the association between an *IllustratorNode* and a *3D-Object* in Figure 4) and allows an *IllustratorNode* to adapt the appearance of the graphical counterpart.

All information for the placement of an *IllustratorNode* is encapsulated in the *ZoomNode* (the position, the scale factor as a request to the *Zoom*). The separation between zoom-specific (or more general layout-specific) information and application-specific information (managed by an *IllustratorNode*) has proven to be very useful for modifications.

Representations of different levels of textual description

An important issue in the design of the ZOOM ILLUSTRATOR is the concept of *representations*. Each node has several representations where the set of representations varies depending on the node's category. Each node has at least a label and it may have explanations. In our domain, anatomy, we categorize nodes as to their membership to an organ system, e.g. bones or muscles.

A *Representation* is an abstract class. It is characterized by its *level of detail* and the *neededSize* to be activated (higher levels of detail corresponding to more information and more space needed). Each representation has a *display()*-method to activate itself.

Representations log their activations and adapt the behaviour of their *display()*-method to the occurrence of previous activations. This is useful if several representations have the same *level of detail*. In this case a direct mapping of the DOI to a representation is not possible, instead the most suitable should be selected. The strategies for the selection of representations are described in detail in Rüger *et al.* (1996).

Label and *Explanation* are subclasses of *Representation* which differ in their *display()*-method. The *Label::display()*-method simply decides whether the label string or a shorter string with abbreviations can be accommodated and displays it. *Explanation::display()* arranges a longer text in a rectangle with a given width with bold parts and hyper links to other nodes (*global links*) as well as to other explanations of the same node (*local links*).

Annotating 3D-Graphics

An important issue in generating illustrations which integrate images and text is the annotation of images. It is especially a problem to calculate reference points within an image to which reference lines point. This calculation should fulfil several requirements:

1. The reference point should clearly belong to the object to be annotated. This is not as trivial as it may seem. The bounding box centre or the centre point, which seem to be good candidates may be outside the object if it has a concave shape.
2. The resulting point should be visible, which is even less trivial and implies that the calculation is view-point dependent.
3. Finally, it should be fast. This implies the calculation of reference points for complex objects can not take into account all surface points of this object.

The first requirements can be fulfilled using a vertex of that object. All vertices obviously fulfil the first requirement, and if none of them is visible it is very likely that the whole object is invisible and the second requirement cannot be fulfilled. To ensure that the calculation is finished in a reasonable amount of time (3), the calculation is adapted to the number of vertices of an object (for objects with many vertices only a fraction is tested as to whether they are appropriate as reference points).

The annotation is achieved as a collaboration of an *IllustratorNode* (which holds the coordinates of the label) and the related *3D-Object* which holds the coordinates of the reference points.

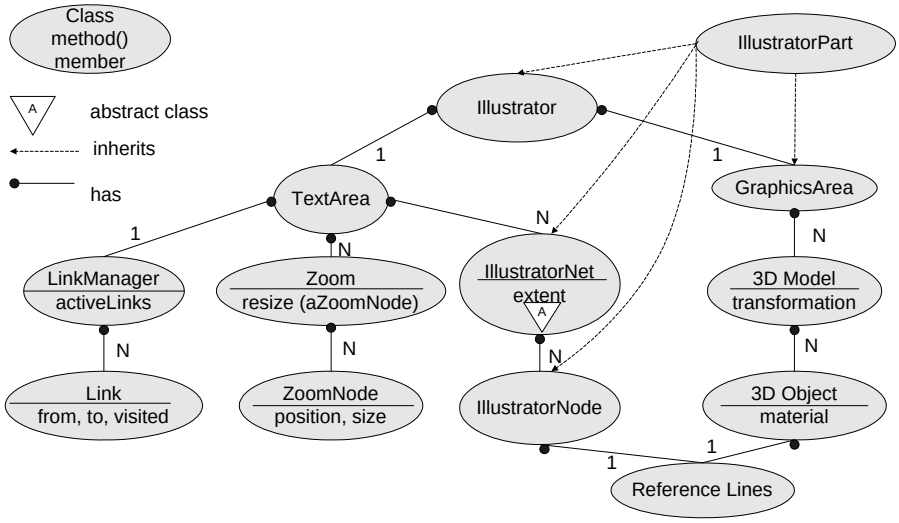

Figure 4: Relations between the most important classes, notation according to Booch (1994)

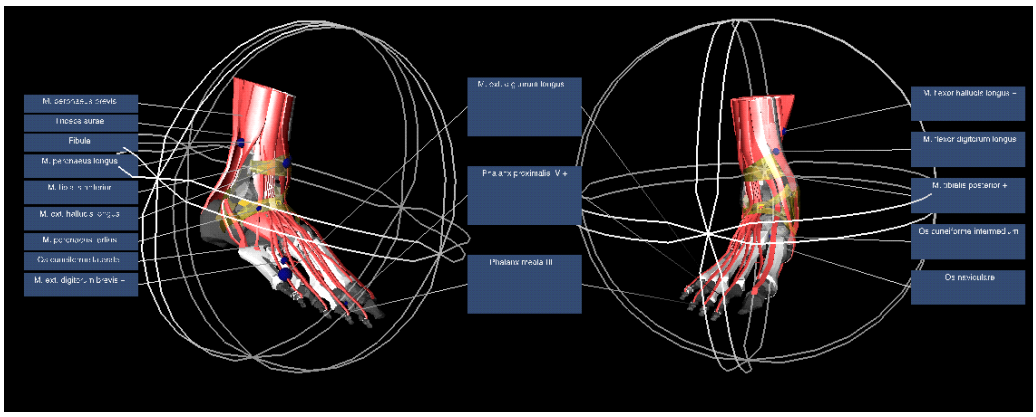

Figure 5: Independent handling of several instances of a 3D-Model using Trackball-Manipulators. Labels which refer to objects which are visible in both instances are placed automatically between them and connected to both 3D-Models.

5 DESCRIPTION IN TERMS OF A MODEL FOR IMMP SYSTEMS

Our system targets at interactive behaviour and even those features of the system which may sound "intelligent" at the first glance, like the selection of representations, are in fact not. Instead a simple comparison between numerical values takes place. In a system where complex 3D-models are interactively transformed, this is necessary just to keep the response rate reasonable. Although the ZOOM ILLUSTRATOR performs some kind of automatic support (annotation, layout, coordination), it is rather a (simple) Multimedia Presentation System than an "intelligent" one.

Despite of this difference we attempt to characterize the ZOOM ILLUSTRATOR within the terminology of the reference model suggested by Ruggieri *et al.* (1996). It turns out that this is feasible and (at least for the author) interesting. According to the structure of the reference model this description is divided into two parts: The first describes which components of the reference model exist in our system and the second their relations in an architectural scheme which is more general than the one presented in Figure 3.

5.1 Important Terms

Media

The ZOOM ILLUSTRATOR includes written text and graphics which is generated when requested. Output media supported include the screen and printers, where an Offscreen-Renderer generates a postscript-file with the rendered presentation.

Goals

Goals to be achieved include textual explanations (*explain* anObject| aGroup *under* anAspect), viewing from arbitrary directions (*show* aModel *from* directionA {*and from* directionB}. These goals are not directly specified by a user, instead he or she invokes commands via mouseclicks at hyperlinks or via a manipulator (recall Figure 5).

The explicit formulation of these goals, however, is a prerequisite for constructing scripts which are interpreted – resulting in commands to the ZOOM ILLUSTRATOR to build up an animation sequence. The extension of the interactive system to a semi-interactive tool, the development of a scripting language and its combination with the interactive kernel is described in Preim *et al.* (1996).

Presentational commands

To enable the user to achieve these goals, interaction facilities on the textual side as well as for the manipulation of the graphics are provided. When mouse-based invocation of commands does not seem to be reasonable, menu items are provided, e.g. to add/remove an instance of the *3D-Model*, to rearrange labels after geometric transformations.

Application

The external data sources for our system are the textual information and the scene description (recall Figure 3). The scene description describes the geometry of the underlying 3D-model. The textual information consists of two parts:

- Structure information

  ⇒ Existing categories and subcategories
  (e.g. in anatomy muscles, bones and nerves) where subcategories summarize nodes of a category which belong to a certain region (e.g. face muscles)

  ⇒ Linkage of nodes via hyperlinks

  ⇒ Nodes and representations which belong to a (sub)category

- Textual Information for each node structured according to the first part.

The separation of structure information and textual information enables us to decide quickly which nodes are important to tailor the presentation for a specific purpose. Structure information is used to find out which items of the textual information are important with respect to a goal specified.

Knowledge

The ZOOM ILLUSTRATOR records requests to change the presentation of a node (Zoom In, Follow a link). This information is made explicit as a history which contains items of the form (<*node*>, <*representation*>) and furthermore exploited to guide the presentation (recall Section 3). In the terminology of the reference model, this information belongs to the *Discourse Model*. Changes on the graphical part are also recorded and thus the whole illustration can be reconstructed, which allows to return to arbitrary points in the history of interaction.

Design Knowledge

Design knowledge is included in the software, but not explicitly formulated. The required knowledge can be categorized in knowledge on how to focus on pieces of information, how to relate graphics and text to each other and on how to make changes smooth. Smooth and incremental changes are important for both the system (prevent unnecessary rendering) and even more important for the user (care for animated movements instead of rapid changes).

To make geometric transformations smooth, knowledge about rendering parameters, their influence on the quality of the image and on rendering times is required. The reference

model refers to this knowledge as *Media Specific Design Knowledge*. The basic principle is to find a trade-off between quality (essential for looking in detail) and the frame-rate (essential when transforming the 3D-model in which case the response-rate is more crucial than quality). The incorporation of this knowledge is crucial for the acceptance of the system, however it depends strongly on a specific environment and thereby less general than the *Design Knowledge* incorporated.

5.2 Architecture in terms of the Reference Model

The description of the ZOOM ILLUSTRATOR in terms of the reference model, reveals that two important parts are not present (see Figure 6). On the part of the layers involved, the *Content Layer* is missing. A selection of media and of strategies to co-ordinate does not exist. The two basic media are always employed, their coordination is defined by user-defined options and changes in one medium are propagated to the corresponding part in the other medium. Due to the interaction facilities offered and the narrow target area, to illustrate complex spatial phenomena, a planning scheme on how to present something is less important than for systems with a broader scope, like WIP (recall Wahlster *et al.* 1993).

On the other hand, there is no *User Expert* in the scheme, meaning that information concerning the user is not stored. To tailor the presentation, the user is offered *Display Options* (including fonts and colours), and *Rendering Options* (concerning the compromises between quality and frame-rate, forcing different rendering algorithms to apply).

Application Expert

The *Application Expert* contains structural information about the domain, including categories (e.g. muscles), aspects of textual descriptions for nodes of a layer and "knows" which objects are visible from which directions (visibility information derived from 3D-model). The *Application Expert* „knows" how to present objects of a category graphically (colours and material properties) and on standardized viewing directions.

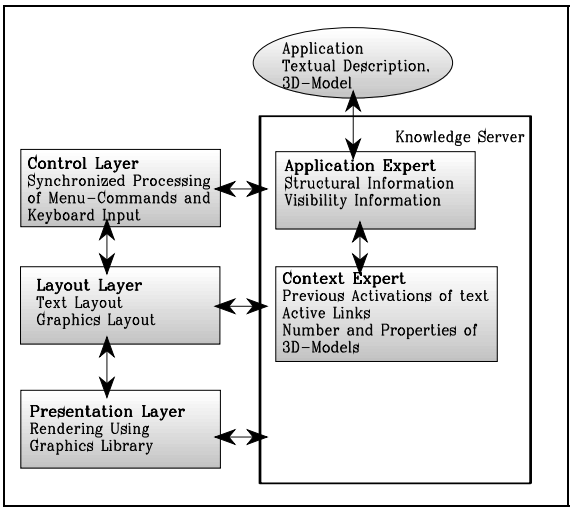

Figure 6: Architecture in terms of IMMP

Context Expert

The design of a *ContextExpert* for our system is strongly influenced by the Reference Architecture. It contains information on active links, on previous activations of textual information, which is exploited to adapt the presentation (e.g. representations to activate, colours). Furthermore information about the state of the graphics is included. For each *3D-Model* the transformation and the state of manipulators is registered, which enables us to save a „homePosition" to which the user may return. For each *3D-Object* the *Context Expert* stores its material and the draw-style (wireframe, filled).

The *Context Expert* is informed whenever an event occurs which changes the presentation and stores it into a list. Each event has an indication whether the user has requested it interactively or whether it is performed by the system automatically as a side-effect, e. g to adapt image and text to each other.

We are currently investigating the generation of descriptive figure captions, which describe the image generated verbally. These descriptions include the current viewing direction, the usage of graphical techniques. Figure captions – as can be found for example in textbooks – are based on the information maintained by the *Context* Expert and comment especially those changes which have been performed by the system automatically.

Layout Layer

The Layout Layer (corresponding to the *Layout-Manager* in Figure 3), is not a central instance, managing all layout problems. Instead instances of each class (recall the class structure in Figure 4) manage their own layout and distribute the space for subordinate instances.

With this policy class *Illustrator* knows how many *3D-Models* and *Networks* are present and assigns space to them. This assignment, includes a rectangular area for usage and a tolerance area which can be used if necessary without informing the superior instance. Only if an instance cannot cope at all with the assigned space, the superior instance (see the *has-a*-relations in Figure 4) is asked to redistribute space. This hierarchical mechanism turns is effective (in terms of speed) and flexible.

An example may clarify this strategy. If the user interacts intensively with textual information at one side, requesting explanations, the network is allowed to "grow" a bit and the nodes involved may even grow a little bit more (recall Figure 1 with the different width of nodes). This can even result in small overlaps between text and graphics display. This is tolerable because overlaps are seldom and the exact extent of the graphics is – due to its irregular shape – difficult to consider. If some limit in the growth of a net is reached and information can only be presented at the expense of others, the user recently interacted with, the *TextArea*-instance is informed. To circumvent the problem, it can move a node to another network or extend the network. Such a change, however, is not incremental and should not occur often, because it is expensive for the system and irritating for the user.

6 IMPLEMENTATION

The implementation was carried out on medium range Silicon Graphics Workstations. This platform allows to experiment with models of a "realistic" complexity, but on the other hand still forces considerations on the efficiency of algorithms. The system uses Open Inventor™, an object-oriented graphics library which targets at interactive applications. For our purpose several characteristics of this library are important: It provides an extensible class library and includes classes for event-handling, interactive manipulation of objects, classes to control the behaviour of objects.

7 CONCLUDING REMARKS

The design and architecture of the ZOOM ILLUSTRATOR have been demonstrated. Whenever possible, the architecture is generic so that at least classes on a higher level should be reusable for similar purposes.

Our system combines interaction facilities with automated techniques to allow more high-level-interaction. This automatic support is especially useful for the annotation of objects, the basic layout and the coordination between images and text.

The architecture of the ZOOM ILLUSTRATOR could be described in terms of the reference model although only a subset of the terms and components have a counterpart in the system. The interaction with textual information could be easily described.

OO-Design is well-suited to state "who" is responsible for which behaviour and to define relations among classes. Furthermore, it naturally leads to an OO-Implementation. The OOD usually results in a vertical structure with hierarchical relations. However, OO-Design tends to bring up lots of classes the overall structure of which is easily lost, especially when little *has-a* or *inherits* relations exist between them.

The horizontal layer structure of the reference model has advantages to explain this overall structure. This becomes obvious, for instance, in the description of the layout policy. The distribution of the responsibility to different classes is very useful for the implementation and maintenance of the software. The overall strategy, however, can be better described within the Layout Layer of the reference model. While the experts presented in the Reference Model lend themselves to be designed as classes in a OO-system, the layer structure is orthogonal to object-oriented design. The tasks being supported by different layers are distributed to many classes in an OO Design. In fact, each part of an illustration performs its own layout calculation and presentation. However, even the layer structure can be exploited to enhance an OOD, because it allows to structure the methods of the classes as to the layers for which they perform a service (many classes have *presentation*-methods, *layout*-methods,...).

Future Work

The architecture presented encompasses text and graphics. Text presented can be clearly assigned to the parts of the graphics it refers to via reference lines. The disadvantage, however, is that both images and text must be processed by our visual system. The ZOOM ILLUSTRATOR would benefit from the incorporation of speech output, which is especially suited for verbal explanations which are not as closely connected to the graphics.

The Reference Model gives us new insights in the system and reveals room for improvement. The inclusion of a User Expert for example would enable us to tailor the presentation to a specific user. Such a personalization is extremely helpful, if the educational aspect of the system is extended (the systems asks questions and answers them).

ACKNOWLEDGEMENT

My thanks go to Thomas Strothotte who has encouraged this work and has initiated my interest in Fisheye Techniques. The author wishes to thank Alf Ritter, who implemented the initial version of the ZOOM ILLUSTRATOR. Furtehrmore my thanks go to Knut Hartmann for commenting on the paper.

REFERENCES

André, E. and T. Rist (1994)
"Multimedia Presentations: The Support of Passive and Active Viewing", *Working Notes of the AAAI Spring Symposium on „Intelligent Multimedia Multimodal Systems"*, Stanford, Mars, pp. 22-29

Booch, G. (1994)
Object-Oriented Analysis and Design with Applications, The Benjamin Cunnings Publishing Company, Second Edition, Redwood, California

Dill, J., L. Bartram, A. Ho, and F. Henigmann (1994)
"A Continuously Variable Zoom for Navigating Large Hierarchical Networks", *Proc. of IEEE Conference on Systems, Man and Cybernetics*, November, pp. 386-390

Furnas, G.W. (1986)
"Generalized Fisheye Views", *Proc. of ACM SIGCHI'86* Conference, Boston, April, pp. 16-23

Noik, E.G. (1993)
"Exploring Large Hyperdocuments: Fisheye Views of Nested Networks", *Proc. of ACM Hypertext and Hypermedia*, Seattle, November, pp. 192-205

Preim, B., A. Ritter, and T. Strothotte (1996)
"Illustrating Complex Phenomena: A Semi-Interactive Approach", *Proc. of Visualization in Biomedical Computing*, Hamburg, September, pp. 23-32

Rist, T., A. Krüger, G. Schneider, and D. Zimmermann (1994)
"AWI – A Workbench for Semi-Automated Illustration Design", *Proc. of Advanced Visual Interfaces*, Bari, Italy, May, pp. 59-68

Ruggieri, S., M. Bordegoni, G. Faconti, T. Rist, P. Trahanias, and M. Wilson (1996)
"Intelligent Multimedia Presentation Systems – A Proposal of Reference Model", Council for the Laboratory of the Research Councils, *Technical Report RAL-TR-96-011*, February

Rüger, M., B. Preim, and A. Ritter (1996)
"Zoom Navigation: Exploring Large Information and Application Spaces", *Proc. of Advanced Visual Interfaces*, Gubbio, Italy, May, pp. 40-48

Seligmann, D. and S.K. Feiner (1991)
"Automated Generation of intent-based 3D-Illustrations", *Computer Graphics 25(4)*, Chicago, July, pp. 123-132

Wahlster, W., E. André, W. Finkler, H-J. Profitlich and T. Rist (1993)
"Plan-Based Integration of Natural Language and Graphics Generation", in *AI-Journal* 63, pp. 387-427