In Proceedings of the INTEGRATED DESIGN & PROCESS TECHNOLOGY "IDPT'99", June 1999. to appear in the IDPT 2000 proceedings.

A Component-based Construction Kit for Algorithmic Visualizations

Abdulmotaleb El Saddik[1], Cornelia Seeberg[1,2], Achim Steinacker[1], Klaus Reichenberger[1], Stephan Fischer[1] and Ralf Steinmetz[1,2]

1
Industrial Process and System Communications
Dept. of Electrical Eng. & Information Technology
Darmstadt University of Technology
Merckstr. 25 • D-64283 Darmstadt • Germany

2
GMD IPSI
German National Research Center
for Information Technology
Dolivostr. 15 • D-64293 • Darmstadt • Germany

{abed, cornelia, stein, reichen, fisch, rst }@kom.tu-darmstadt.de

ABSTRACT

In this paper, we describe a component-based architecture which allows to create interactive teaching applets simplifying the understanding of complex technical processes. In contrast to existing approaches the user can experiment interactively with components thus influencing the presentation and the results of the algorithms being illustrated. We explain how applets can be created using modular units (ItBeans), how the user can combine these using the Interactive Teaching Bean construction Kit (ItBeanKit), which allows users to create Interactive Visualization Artifacts (IVAs), according to a desired level of functionality and appearance, suitable for their specific needs.

1. INTRODUCTION

In the past a lot of electronic teaching material including electronic books, intelligent tutorial systems and web-based courses emerged. Most of these also commercially available products use a variety of different media such as video, audio, images, animations and hypertext to exploit the nature of hypermedia at its best in order to yield some sort of optimal learning success.

In addition, a vast amount of animations has been generated, caused by the rapid growth of the WWW in combination with languages like Java and VRML. However, these animations often show some major drawbacks, such as

- *Video-like nature of animations.* Most of the animations being used to visualize complex algorithms and techniques cannot be influenced by the user. A user can only observe the ongoing animation and try to understand the underlying theory. As the only available form of interaction most animations use parameters to change the output.
- *Experiments.* The user cannot change the behavior of applets by omitting certain steps or by adding or exchanging components.
- *Connection of animations.* Most applets available nowadays are running in a stand-alone mode. The user, e.g. cannot connect an animation of a video decoder to that of a network and study the resulting effects.
- *Reusability*. Many animations have been developed without regard to software engineering in terms of reusability. The animation of JPEG [1] or MPEG [2] serves as a good example: Even though both compression schemes use the Discrete Cosine Transform (DCT), and the Huffman encoding, a reuse of a component coming from an already finished animation of JPEG can in most cases not be used to visualize a step of the MPEG-compression process.
- *Hierarchical structure.* Most applets do not deal with changing user requirements. Beginners, as well as intermediate students and experts can nowadays only use the same animation.

It is the goal of the interactive Teaching project (iTeach) of the Darmstadt University of Technology to create a system with an optimal support for the learner, integrating a variety of different learning styles and a component-based applet architecture which can be used to experiment with complex algorithms. As the research area of adaptive hypermedia systems is being explored for a long time now it is the particular goal of iTeach to examine how *multimedia* can be used best to support the learning process thus extending the focus from systems most often based exclusively on text and images to real multimedia systems integrating text, images, audio, video, and animations. This also implies the knowledge which of these media has to be used with respect to the particular

content to be explained. Especially the meaningful use of different media is being neglected in many commercial systems following the paradigm "the more – the better".

In this paper we concentrate on the description of an effective way to create *interactive teaching animations*. Unlike other work, our model takes a user-centered view (*what does the user need to see?*) rather than a designer-centered view (*what can we show him?*), and employs a multilayered presentation to improve the effectiveness of learning. Our approach enables the developer to employ reusable components which can be combined to create a complete animation for a given problem. To simplify this process we use a visual builder tool to create a complete animation from predefined components. We call this framework, which contains images, text, audio, and animation the Interactive teaching component-based (Beans) toolKit ("ItBeanKit").

This paper is structured as follows: In Section 2 we describe the iTeach-project to explain the background of our work. Section 3 examines the interaction aspects we identified. In Section 4 we explain the architecture of our visualization construction kit. Section 5 shows the application of our concept being applied to the compression scheme JPEG. Section 6 reviews related work. Section 7 concludes the paper and gives an outlook.

2. ITEACH-SYSTEM

The iTeach-System (interactive Teaching-System) strives for an efficient learner-support in the process of teaching multimedia and network technology. The system consists of an electronic text book explaining the topics to be learnt [1]. The book is extended by a didactic component which orders the content with regard to specific learning strategies, based on a user profile which guides the user by tracking the actions he is performing when reading the electronic book. The individual chapters of the book are generated on the fly on the basis of the learning style which has been chosen. We currently support hierarchical and constructivistic learning as well as learning based on examples. Hierarchical learning explains the theory of a topic followed by examples and applications. Constructivistic learning starts with applications followed by the theoretical background. Learning by example first presents examples which are then generalized to explain theories in a comprehensive manner. Using an explorative style (learning by example), a user first has to learn a specific subject by using an applet which is not of a static nature in a sense that parameters can be modified and components can be exchanged.

Applets serve as an important concept to illustrate complex algorithms. The iTeach-system uses animations both in the textbook and in a specific learning environment which allows for experimenting with component-based animation of algorithms. The integration into the textbook is done on the basis of a script: the teacher (or author) specifies the components and their order to generate an applet which is then included in the book as a guided-tour-applet. If the user wishes to experiment with the algorithm, the applet-toolkit can be invoked which allows for exchanging components, setting parameters or in the case of mathematical animations specify new functions which are then animated. As we use a generic representation of the components, new components can be included by an insertion into the component-chain. The user could e.g. decide to run a Huffman-encoding followed by an arithmetic encoding which would not be available in a standard applet animating a compression scheme. The main difference between the textbook and the toolkit is that the nature of the applets is static for the guided tour in a sense that the composition as well as the functions of an applet cannot be changed. As applets in the textbook are used to explain exactly the theory described in the text, modifications are only distracting the learner. However, it is always possible to switch between the textbook and the toolkit.

In the next Section the necessary conditions to develop an efficient interactive toolkit are reviewed.

3. ACQUISITION OF INSIGHT

The ability of the human being to acquire insight into multivariate data can be enhanced by the opportunity to view that data in a visualized form. Such a process of visualization can be even more effective if interaction with the data's presentation is supported. When developing a visualization scheme to support a wide variety of users, it becomes apparent that there are a number of facets which should be taken into consideration:

- Identifying the needs of the user.
  What do users want to see?
- Establishing useful and meaningful representations
  What is the best representation to convey the information clearly, concisely and efficiently?
- Investigating the problems of scale.
  Meaningful visualization of both single components and variable sized collections of components are needed.
- Introducing flexibility.
  Techniques enabling the user to customize and control the visualization should be investigated.

Our aim is to provide a useful and meaningful visualization to assist in the learning process of algorithms. The effectiveness of visualizations can be enhanced if a collection of components are provided from which applications can flexibly and fluently be created. Such a proposition, which is the subject of this paper,

immediately identifies the issues *interaction* and *support* of the learner.

Interaction implies that the learner is guided in the sense that he can get feedback if problems emerge. Assuming that the handling of the toolkit itself is intuitive such problems can only result from the difficulty of the topics to be learned. The *difficulty* of an algorithm to be animated can result either from the knowledge of the learner which might not be sufficient to understand the topic or from the amount of information presented by the animation. If the user's knowledge is not sufficient to understand parts of an algorithm we offer two possibilities to acquire the corresponding knowledge: a user can read a short explanation of the part of the algorithm he currently executes or he can invoke the chapter of the textbook explaining the underlying theory in depth. The latter includes search functions to get a more specific way of explanation.

The processing of an insufficient knowledge of a learner is performed in a traditional way by using hyperlinked multimedia documents. The second problem however, the density of the presented information has to be dealt with by another approach, the use of *levels of complexity*. The idea behind a level of complexity is that a user can reduce the information density of a part of an algorithm by splitting the part of an animation he/she is currently using into a number of steps which can be understood easier equivalent to a smaller information density. This process is shown in Figure 1. While *C* stands for complexity, the upper index denotes the level, the lower the number of a component.

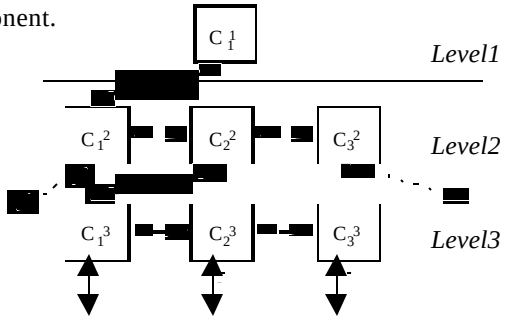

Fig.1: Levels of complexity

The particular components of the levels can be implemented as JavaBeans [3] [4] [15]. Looking at the animation of JPEG (section 5.2) there might be one component in the first level only showing the resulting image and data like the file size of the compressed as well as the uncompressed image. In the second level there could be the four steps: preparation – DCT – quantization – entropy encoding. The third level could explain the DCT as well as the different algorithms available to perform the entropy encoding. We distinguish between three kinds of interaction the user is provided with:

1. Variation of parameters of a running itBean
   - Visualization (level of complexity)
   - Animation (speed, background color, foreground color,..)
   - Simulation (interaction by the user interface)
2. Composition and substitution of itBeans
   - Example: Substitute DCT (Discrete Cosine Transformation) by FFT (Fast Fourier Transformation) to modify the existing JPEG processing.
3. User guidance
   - Help function
   - Guided tour
   - Step function

4. VISUALIZATION CONSTRUCTION KIT

A major advantage of a visualization construction kit is that the composition and function of the eventual interactive visualization artifact is under the control of the user who is thus free to select and combine components such as data producer and encoding blocks according to his needs. Fischer and Lemke [5] describe such a construction kit as "a set of building blocks that model a problem domain. The building blocks define a design space, that is, the set of all possible designs that can be created".

The ItBeanKit currently being developed provides the user with a workspace where visualization components can be introduced, linked and where interaction can take place [see figure 5]. When executed, all the components are contained within windows, thereby facilitating their easy resizing, placement and closure. An important aspect in our framework is substitution, which means that an end-user can exchange an ItBean with another. In the context of JPEG the exchange of DCT with FFT may be an appropriate substitution.

ARCHITECTURE OF THE TOOLKIT

The architecture of the itBeankit is shown in Figure 2. To be able to coordinate different itBeans (Interactive Teaching Beans) we use a main controller called the it-Main-Controller. the itBeans themselves are developed according to the *model-view-controller* model (MVC-model) [6] which is responsible to provide a unified environment of the itBeans, which do not have to care about the graphical presentation of their input and output. We now explain the components of our architecture in detail.

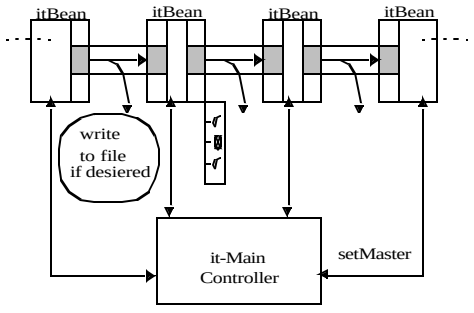

Fig. 2: Architecture of itBeankit

ITBEANS

ItBeans are the major components responsible for the animation of parts, or of the whole of the algorithm to be explained. ItBeans consist of a set of state and shared attributes, a set of elements responsible for the creation of the user interface and also of a set of methods to define relationships and interdependencies among state/shared variables and events. The external control over an itBean is performed by the controller which is explained below. An itBean itself can therefore only work in a single level of complexity. The switching between these levels is done by the controller unit.

An important property of the itBeans is that the user can forgo a graphical presentation of the results of the calculations an applet executes. The visual representation can thus be turned off at runtime but still be executed inside the framework. Looking at the JPEG-compression this feature can be used if a user is particularly interested in the effects and in the function of the quantization step. He then turns off the graphical representation of the image preparation and of the transformation (DCT). While these steps are still executed and thus the content of an image is modified, all the user will see is the animation of the quantization. In our toolkit we provide buttons to turn on/off the representation of certain steps of the algorithms to be animated.

Another main aspect in our implementation is the internationalization [7]. Each of our ItBeans supports English, German and Spanish. The support of new optional languages is guaranteed due to the Java Internationalization API [8].

Fig.3: ItBean Internationalization support

IT-MAIN-CONTROLLER

The main controller is responsible for the invocation of the proper itBean corresponding to an event. Our controller currently reacts on the following events: *play*, *stop*, *hide*, *next*, *previous*, *level down* and *level up*. *Next* and *previous* could also be implemented without a controller as these events specify the itBean which the user wants to be executed after the one being currently active. This could also be controlled within the respective bean. The *level*-commands are responsible to traverse upwards or downwards the hierarchical tree of parts of the algorithm to be shown. To simplify the interaction with the user interface only those buttons which are applicable are highlighted. Using the components at the lowest level of the hierarchy only the *level up*-button is highlighted. An example of the user interface of the components is shown in Figure 4.

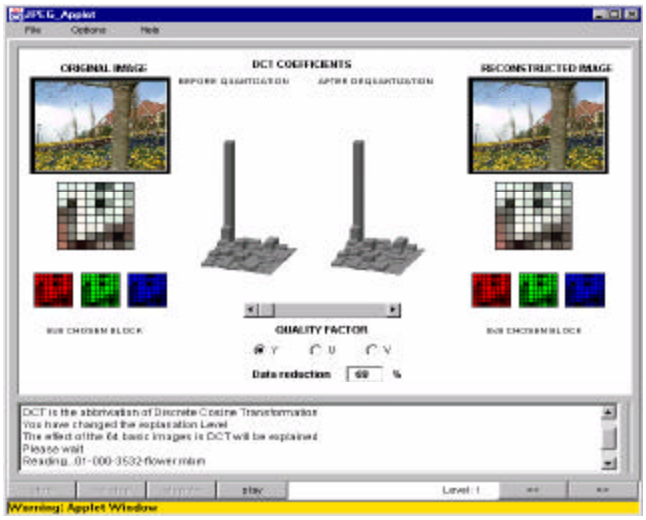

Fig.4: User interface of component beans. Buttons fire events which are passed to the controller

COMBINATION OF ITBEAN-COMPONENTS

The combination of itBeans can be accomplished in two different ways: itBeans can be combined a) using the visual builder tool or b) writing a script specifying the components and their respective order for the purpose of creating an applet. Figure 5 shows the prototype of the visual builder tool, which allows a simple wiring of the components to be tested. The visual builder tool also allows to generate an applet automatically with its corresponding web page (HTML-file).

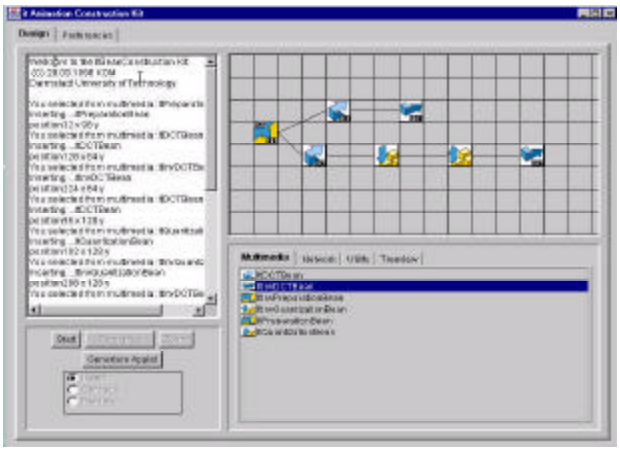

Fig.5: The prototype of the ItBeans visual builder tool

5. CASE STUDY: ANIMATION OF JPEG

In this section we demonstrate the use of our toolkit. As an example we present the visualization of the different steps of the JPEG compression, including the procedures being executed at the encoder as well as those of the decoder [1]. A JPEG encoder performs a series of transformations on raw image data to produce a compressed output stream, which is then transmitted to a JPEG decoder that executes these transformations in the inverse order to decompress the image. Figure 6 outlines the steps of the JPEG compression/decompression process.

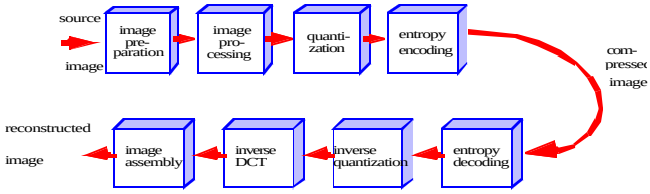

Fig. 6: The JPEG compression/decompression process

JPEG deals with colors in the YUV color space. For each separate color component (Y, U and V), the image is divided into 8x8 pixel blocks of picture elements. Each block is then transformed into a two dimensional DCT matrix. Figure 7 shows an 8x8 coefficient matrix generated by the DCT step. Coefficients with lower frequencies (typically higher values) are encoded first, followed by higher frequencies (with typically small values, near to zero).

The coefficients of the DCT matrix are then quantized. The final step of the JPEG compression consists of an entropy coding.

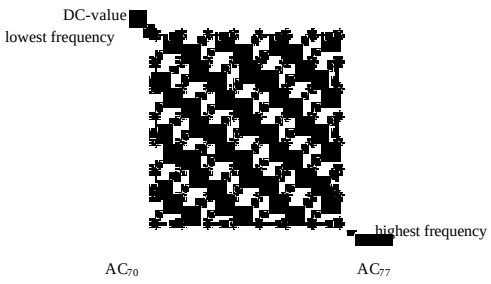

Fig. 7: DCT processing order

To develop an applet explaining the JPEG encoding and decoding process, a set of reusable itBeans that perform these transformations are composed as illustrated in Figure 8. The ItBeanKit library uses a fine grained modular decomposition to effectively de-couple the visualization of each step of the algorithm. This concept allows for a significant reuse of code when developing new components, and moreover some of these can directly be reused to compose other software components. Although, the identification of the components necessary to visualize a complex algorithm is performed applying a top-down technique, the development of these components (ItBeans) follows a bottom-up design. The development of the ItEncodingBean serves as a good example how this is achieved: no new code visualizing the encoding process has to be developed. Instead of implementing a new component, the functionality of one of the different encoding algorithms can be reused.

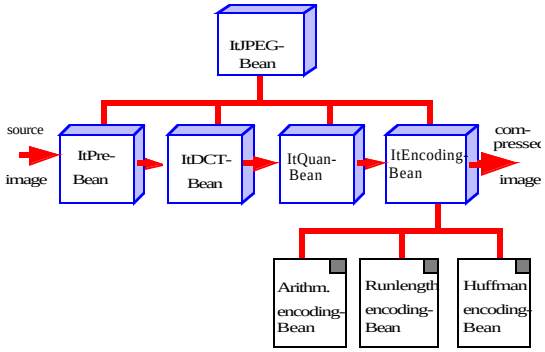

Fig. 8: Bean components of JPEG compression process

Each of our itBeans can have more than one view with respect to the specific needs of the user. An end-user who does not need to be concerned with the details of the DCT transformation algorithm or the Huffman encoding, but who is only interested in the general functionality of JPEG, will get the ItJPEG-Bean presenting the input and the output image with some additional information like the file size before and after the compression. Another end-user who might be interested in the execution of the DCT transformation algorithm will get the ItBeans which are located at the second level with regard to our hierarchy.

JPEG IMPLEMENTATION

In the following we will explain our JPEG implementation with all the necessary steps according to our framework. In order to develop an applet explaining the JPEG encoding and decoding process, a set of reusable ItBeans that perform these transformations are composed as illustrated in Figure 6.

As mentioned earlier the ItBeanKit library uses a fine grained modular decomposition to effectively decouple the visualization of each algorithm. This allows a significant reuse of code when developing new components, and moreover some of these components can directly be reused to compose other software components.

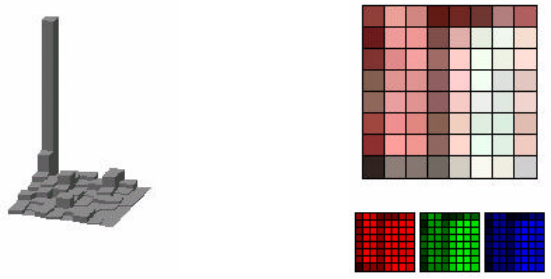

Fig. 9: 3D view-Bean and RGB-Bean

The 3D view-Bean which shows the values of the Discrete Cosine Transformation, as well as the RGB-Bean, which is shown in Figure 9 are used in different applets (for example JPEG, DCT and the preparation unit applet).

As shown in Figure 4, (JPEG applet) the user can choose an image block and see the composition of the block in RGB (Red-Green-Blue) colors. The user can also explore the quality of the picture by choosing a higher quantization value which then decreases the quality of the picture by increasing the compression factor and decreasing the file size of the picture. If the user presses the forward button, an illustration of the steps of JPEG will be provided. By clicking on such a component a new applet window is opened and the detailed step is explained. Figure 10 illustrates the DCT visualization, having clicked on the DCT Button. As illustrated the user can switch between FDCT (Fast Discrete Cosine Transformation) and DCT (Discrete Cosine Transformation), to explore the time differences necessary to calculate the output values (DCT is outperformed by FDCT). The user can also change the parameters of the formula, and define his own cosine function to be executed. In DCT level 2 the user can explore the effect of the 64 basic frequencies. He / She can interact with the system and examine the processing order of the AC-coefficients using the zig-zag sequence described above. Furthermore Figure 10 illustrate how also how a visualization of the RGB-Bean is reused.

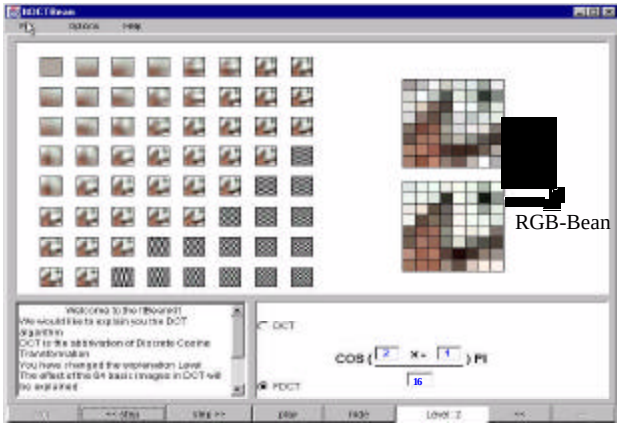

Fig. 10: Illustration of the DCT applet in level 2

6. RELATED WORK

The work described in literature shows that the use of component based architecture can significantly increase the learner's efficiency. Stasko et al. [9] studied how students learnt from an animation of computer algorithms, and concluded that they only learnt from the animation if they constructed it themselves. Similarly, Nardi and Zarmer [10] carried out a study of spreadsheet users and observed that these users tended to "incrementally develop external, physical models of their problems".

Klein and Hanisch describe a system to realize an interactive computer graphics course [11]. Their system employs a similar concept to use Beans to visualize different steps of a complex algorithm. However, no control structure is provided to group modules and to switch between different complexities.

Wernert chooses a similar approach for a unified environment for the presentation, development and analysis of graphics algorithms [12]. His system also does not use the concept of complexity. Furthermore it only works on the IRIS explorer.

Mecklenburg and Burger describe a system to create component-based applets automatically from Estelle-specifications [13]. They also plan to use complexities to support the learner. Their approach is somewhat different as the order of the components is bounded by the Estelle-specification.

Land [14] describes a similar approach as [12] allowing for high-level programming (network wiring). As in [12], his system cannot be interlinked with a WWW-based environment.

7. CONCLUSION AND OUTLOOK

In this paper we described a component-based architecture for animations using JavaBeans. Our approach extends other concepts by the use of hierarchies thus supporting the learner in an efficient way. Instead of merely wiring components we propose to use a controller unit to switch between aggregated and detailed views of the steps of an algorithm to be animated. The different steps of an algorithm are animated by particular itBeans. To achieve a common look-and-feel we separate the graphical output from the itBean itself.

Our experience suggests that the ItBeanKit may well provide an environment in which a user without conventional programming skills can build a useful interactive visual algorithm relevant to a particular task. The ItBeanKit will therefore continue to be extended, particularly by increasing the available choice of components.

At the moment we study the extent, by which algorithms can be modified. In this paper we looked at the animation of JPEG. In the near future we will examine how routing in the Internet can be animated. We will try to build up a hierarchy which has routers, links and end-systems at its lowest level.

Furthermore we will study how the itBeankit can be integrated in a distributed learning environment enabling students who work in a distributed environment to discuss the functionality of algorithms. In addition we examine new forms of tests. Instead of using multiple choice students can experiment with components. If these are given the student has to prove his knowledge by finding the right order in which the components have to be placed. It is thus immediately possible to observe if the answer is correct or not because the result is presented graphically. If a learner places the entropy encoding of the JPEG-compression in front of the DCT, the result will change significantly.

8. ACKNOWLEDGEMENT

The authors would like to thank the Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) which partially funded the research project and the Volkswagen Stiftung. We owe also thank to our colleague Lilo Kolb and Moni Jayme for their constant support.

9. REFERENCES

[1] Steinmetz Ralf, and Nahrstedt Klara: *Multimedia computing, communications, and applications.* ISBN 0-13-324435-0, Prentice Hall 1995.

[2] Joan L.Mitchell, William B. Pennebaker, Chad E. Fogg, and Didier J. LeGall: *MPEG video compression standard.* ISBN 0-412-08771-5, Chapman &Hall 1997.

[3] Hamilton, G.: *The JavaBeans API specification*. Sun Microsystems, 1997.

[4] M. Morrison, *"JavaBeans"*, Sams net, ISBN: 1-57521-287-0, 1997

[5] Fischer G. and Lemke A., *"Construction Kits and Design Environments: Step Toward Human Problem-Domain Communication",* Human Computer Interaction, Volume 3, 1987

[6] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *"Design Patterns - Elements of reusable Object-Oriented Software"*, Addison Wesley ISBN 0-201-63361-2, 1995

[7] John O'Conner: *Java Internationalization: An Overview*, http://developer.java.sun.com/developer/technicalArticles/intl.html

[8] Flanagan, David: *"JAVA IN A NUTSHELL``*, 2nd Edition, O'Reilly, ISBN 1-56592-262-x, 1997

[9] Stasko J., Badre A. and Lewis C., *"Do Algorithm Animations Assist Learning? An Empirical Study and Analysis"*, ACM Interchi Conference Proceedings, 1993.

[10] Nardi B.A. and Zarmer C.L., *"Beyond Models and Metaphors: Visual Formalisms in User Interface Design"*, Journal of Visual Languages and Computing 4 , 1993.

[11] Klein, R. and Hanisch, F.: Using a modular construction kit for the realization of an interactive Computer Graphics course. In proceedings of EdMedia, 1997.

[12] Wernert, E.: A unified environment for presenting, developing and analyzing graphics algorithms. Computer Graphics, 31(3), 1997

[13] Burger, C., Rothermel, K. and Mecklenburg, R.: *Interactive Protocol Simulation Applets for Distance Education*. To app. in proceedings of IDMS98, Oslo, 1998.

[14] Land, B. R.: *"Teaching computer graphics and scientific visualization using the dataflow, block diagram language Data Explorer"*. In proceedings of the IFIP WG 3.2 Working Conference on Visualization in Scientific Computing, Uses in University Education, Irvine, USA, 1994.

[15] Joseph O'Neil: *"JavaBeans Programming from the GROUND UP"*, Osborne, ISBN: 0-07-882477-x, 1998

[16] David M. Gery, Alan L. McClellan: *"Graphic Java - Mastering the AWT"*, Prentice, Hall ISBN 0-13-565847-0, 1997