Journal of Universal Computer Science, vol. 7, no. 5 (2001), 420-433
submitted: 14/4/01, accepted: 7/5/01, appeared: 28/5/01 © Springer Pub. Co.

Animations for Teaching Purposes: Now and Tomorrow

Tobias Lauer
(University of Freiburg, Germany
lauer@informatik.uni-freiburg.de)

Rainer Müller
(University of Freiburg, Germany
rmueller@informatik.uni-freiburg.de)

Thomas Ottmann
(University of Freiburg, Germany
ottmann@informatik.uni-freiburg.de)

Abstract: Animation is commonly seen as an ideal tool for teaching dynamic phenomena. While there have been very few studies testing this hypothesis, animations are used extensively in teaching, particularly in the field of algorithms. We highlight features that we consider important for animation systems, describe the development of algorithm animation by examples, and present a new Java-based system supporting annotation and recording of animations. We also outline a way to annotate animations and movies given in the MPEG video format. By listing several case studies we describe new ways and possibilities of how animation systems may be used in the future.

Key Words: Algorithm Animation, Animation System, Annotation Capturing, Education, Multimedia Authoring, Presentation Recording

Category: Computing Milieux – Computers and Education – Computer Uses in Education (K.3.1)

1 Introduction

It is quite common to promote algorithm animation with the argument that an animation is the best way to explain a dynamic, i.e., a time-varying, phenomenon. Instead of drawing by hand a series of pictures of the various stages of a running algorithm, it is certainly more convenient to run the algorithm on a computer and have an animation system generate the series of figures. This way, sorting and searching algorithms, graph algorithms, algorithms for processing text, and many others have been extensively used in courses on algorithms and data structures.

Whether or not the animation of an algorithm is appropriate or well-done is difficult to judge and quite often just a question of personal taste. The relevant question in this context is, however, whether an animated algorithm can provably enhance knowledge acquisition. This can be answered only through carefully designed comparative studies and it may even depend on the specific subject. There are only very few studies of this type currently available. Hence, it is not quite clear whether teaching the design and analysis of algorithms by using classical means like blackboard and chalk, and (maybe) transparencies, is less effective than the modern tools, computers and animation systems that are available today. In order to understand an algorithm, learners must ultimately generate their own mental images and "movies." Therefore, it might even be counter-productive to provide them with computer-generated prototypes!

Nevertheless, it is our belief that animation is a useful tool in education and that the *visual coherence* provided by a smooth animation can, in many cases, visualize dynamic phenomena much better than static pictures.

2 Desirable Features of Animation Systems

In this section, we highlight some important features that animation systems should provide when used for purposes of teaching. In particular, these are the aspects of interaction, annotation and recording. We also address the issues of production and handling of animations.

2.1 Interaction

The usual way of including an animation in a lecture is to run it in *movie mode*: a sorting algorithm is run on a carefully selected sequence of input values, and the critical operations carried out by the algorithm (comparisons and exchanges of keys in the case of sorting) are visualized. The animation may occasionally be paused by the instructor for questions or explanations, or important parts might be repeated for better understanding, but this is what interaction is often limited to.

The real power of computer animation systems, however, is the possibility of interacting not only with the replay device but with the algorithm itself. In the visualization of a data structure, the user might be able to choose the next operation to be carried out, resulting in different possible outcomes of an animation. This kind of interaction may differ in degree, ranging from the simple selection of inputs to comfortably "communicating" with the animation and manipulating objects via mouseclicks. But even running the same algorithm on different sets of data will be far more insightful to students than watching it run several times on the same input. We will give several examples for this kind of scenario in later sections.

Furthermore, interactive animations, or simulations, can be used in a greater variety of learning contexts. While movie-style animations are destined to be used in lecture-style teaching with the students being more or less passive consumers of the delivered contents, animations with a sufficient degree of interaction will enable the learners themselves to experiment with the algorithm. This can be used in labs accompanying lectures or in students' own studies at home.

The possibility to interact with an algorithm and to observe its behavior is considered crucial for self-paced learning and studying. As has been pointed out in one of the few studies carried out in the field [Kehoe et al. 1999], it is exactly this use of animations which has the most significant advantages in respect of learning success, compared to traditional teaching methods such as static pictures or plain text.

2.2 Annotation

In lectures and presentations teachers usually annotate the documents they present. This is done mainly in order to highlight important features, fill gaps, or include additional information.

Animations could certainly benefit from annotation as well. It would be very convenient for an instructor to be able to write or draw directly onto the graphical output of the animation.

This can be achieved by providing an additional, transparent "layer," on which all annotated material is drawn. The annotations themselves would be done with the mouse or a pen in front of a computer screen or using an electronic whiteboard or touchpad.

It is clear that projecting the animation to a (non-electronic) whiteboard and writing on it with a normal pen will not be sufficient. First, it would be inconvenient to have lengthy breaks during the animation for clearing old annotations. Second, and more important, computer generated animations can allow for annotating specific objects that are part of the animation. Such annotations, for example, highlighting, could then move along with those objects, whereas general information is expected to stay where it is as the animation continues. For this scenario it is required that the animation system itself should support the annotation of animations.

A different situation arises with the annotation of animations or movies in MPEG format. We address this issue in greater detail in [Section 5].

2.3 Recording

Today a growing number of educational courses are recorded or otherwise preserved in order to be used by distant learners or students who have missed a class. With ongoing projects all over the world such as virtual universities, this development can be expected to continue in the future. Consequently, it is desirable that all material, including animations, be captured and integrated in a presentation document recorded from such a session.

Simply videotaping an animation will certainly not be sufficient. First of all, the quality of the resulting movie is in most cases below what is considered appropriate. Second, it seems a shame to have material available in digital form, tape it and then re-digitize the video with the loss of both quality and all the "semantic" information which might have been there in the original. And third, it would be very useful to preserve the interactivity that an animation system may offer [see Müller 2000].

In recording interactive animations several scenarios can be imagined. The result could simply be a video-like clip without any interaction for the end user. In some cases, it might be better if the learner could interactively modify the animated content. Ideal would be a combination of the two, where the recorded animation may be viewed, but could also be interrupted and interactively modified by the learner. In this last case, the recorded session becomes an interactive multimedia learning document that students cannot just watch but experiment with.

2.4 Easy Handling and Production

For a widespread use of animations in teaching, it is important that these can be handled without difficulty by instructors. This in particular includes non-CS teachers, who may be computer literate but not experienced in programming. Therefore, the handling of animations must be intuitively clear and self-explanatory.

Moreover, it is desirable that such animations may be easily developed and produced, even by people with little or no programming skills. No teacher can afford the time for lengthy developing sessions. Thus, animation authoring systems are needed which must be both powerful and easy to use.

2.5 Integration

If animations are to be used in offline multimedia courses for self-studies, it is necessary to integrate individual animations with other material. In order to achieve this, the animation system must provide some means to synchronize animation with other multimedia streams such as audio or video [Müller 2000].

Another prerequisite of integration is flexibility of use. This includes independency of the underlying operating system. For most media types, platform-independent standards have emerged during the past years. For applications, the best way is to implement them in Java. They may then be integrated into platform-independent course documents.

3 Development of Algorithm Animation

We describe some animation systems that have been used in education, outlining their different philosophies and strategies, and comparing them using the criteria listed above. For a more comprehensive overview of animation systems see [Müller 2000].

3.1 Balsa and Interesting Events

The concept of *interesting events* was introduced by [Brown, Sedgewick 1985]. It is used in the *Balsa* system (*Brown university ALgorithm Simulator and Animator*), and has been adopted by its successor, *Zeus*, and many other systems. Its major goal is the separation of algorithm and animation and the specification of a way for linking those two parts.

The programmer of the visualization takes the following steps. First, examine the algorithm for interesting events, i.e., important operations. Second, find a solution to how these operations may be animated best. Third, program the animations. And last, supplement the original algorithm with the animation components specified as abstract descriptions. For legibility of the algorithm source code, Brown suggested keeping all important operations in separate procedures which carry out the manipulations of the data structure as well as the animations. Sometimes, animation components will have to be inserted at positions in the algorithm where no important things happen, just to attract the viewers attention. These, too, will be coded as interesting events.

Balsa is divided into three components. Input generators create input data for the algorithm. The algorithm itself produces interesting events, which are handed to renderers creating views of the animation. The system does not support graphical annotation, but Balsa sessions can be recorded using a scripting language.

3.2 The Path-Transition Paradigm

The *path-transition paradigm* [Stasko 1990] specifies continuous motion in computer animations and has had a major impact on many of today's animation systems.

The paradigm uses the four types *location*, *image*, *path*, and *transition*. A location describes a "point" in an n-dimensional space. An image is the object of the animation. A path consists of a finite series of locations, along which the object will be animated. Finally, a transition links an image with a path and a transition type such as motion, resizing, or color change.

Following the paradigm, any changes of images are done through transitions. This includes discrete changes like the change of an object's visibility.

The *(X)Tango* system (*X-window Transition-based AnimatioN GeneratiOn*) and its successors, *Polka* and *Samba*, as well as many other animation systems follow the paradigm, even if the later systems may modify some details. In *Polka-RC*, for example, paths no longer consist of a finite number of locations but are described by a continuous function. Fast computers can display the animations with more interpolated steps, thus showing smoother transitions.

Neither XTango nor Polka are platform-independent, but there is a Java version of Samba, *JSamba*, for the use in HTML documents. None of the systems supports graphical annotation or recording of animations.

3.3 Java-based systems

In order to be independent of the user's operating system, most of the animation systems developed today are implemented in Java. We list *JCAT* as an interesting example. Using applets, it is designed for web pages. Distributed animations are supported through the Java RMI interface (*Remote Method Invocation*). Thus the teacher can control animations running on students' machines in a *virtual classroom*.

ANIMAL (*A New Interactive Modeler for Animations in Lectures*) [Rößling et al. 2000] is a tool offering a scripting language and a visual component for creating animations. Thus, authors do not need programming knowledge. However, animations authored that way are not linked to any algorithm. Hence, visualizing a run of the same algorithm on a different set of inputs requires the author to make a whole new animation since only predetermined movie-mode animations can be created.

Like most other systems, JCAT and ANIMAL do not support recording or annotation of animations.

4 A System Supporting Annotation and Recording

In order to meet the requirements that many of the existing animation systems leave to be desired, a new system was developed. *JEDAS* (*Java EDucational Animation System*) [Müller 2000] unifies the advantages of platform independence, annotation, recording functions, and easy production.

4.1 Design and Usage

JEDAS is a Java-based system designed for creating, executing, annotating, and recording two-dimensional animations and simulations. One major aim is to offer a comfortable authoring process and the integration of existing algorithms and animations; another one to provide a presentation service to integrate animations in computer presentations, supporting their recording, graphical annotation, and transmission to remote audiences.

Animations are created in Java using the JEDAS class library. Based on the path-transition paradigm, all types of animation can be created easily and without much effort. Details of the semantic linkage of algorithm and animation are completely up to the authors. They can insert animation instructions directly into the algorithm or use interesting events as an abstract layer between the algorithm and the visualization component.

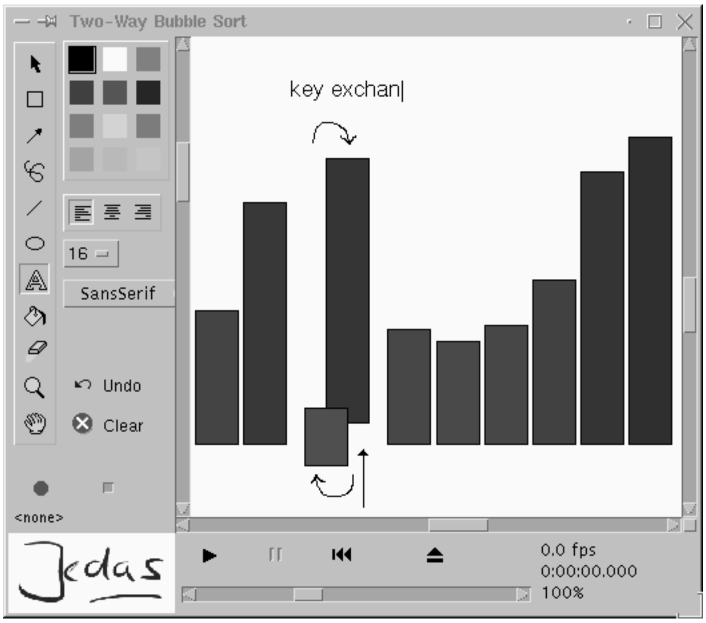

Figure 1: *Annotation of a BubbleSort animation using JEDAS.*

4.2 The Authoring Station

Interaction with the algorithm has to be coded by the animation programmer, but JEDAS offers a control panel allowing the user to start, pause, and reset the animation at any time, and to increase or decrease the speed of the animation. In addition, scroll bars for horizontal and vertical scrolling as well as a zooming function are part of the animation frame.

The JEDAS authoring station (Figure 1) also allows all animations to be graphically annotated; a pointer, simple objects like rectangles and ellipses, as well as text and free-hand drawing are supported. Such annotations can be made at any time, in paused or running animations. In addition to the annotation panel, JEDAS offers a record panel with record and stop buttons to capture an animation including all annotations made during its presentation.

4.3 Replaying Recorded Animations

Annotated and recorded animations can be replayed using the *JedasPlayer* (Figure 2). Since the system supports fast access to any point of a recorded animation (*random real-time access*), random visible scrolling allows the viewer to comfortably navigate within the animation via mouse interaction.

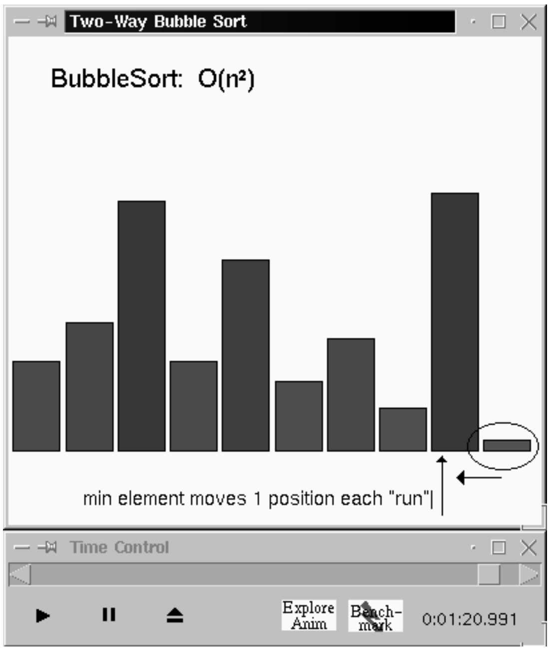

Figure 2: *Replay of an annotated and recorded JEDAS animation.*

Random real-time access also facilitates the synchronization of animations with other media streams for purposes of presentation recording and transmission. These features are used for integrating recordings of JEDAS animations into AOF documents [Müller, Ottmann 2000].

4.4 Animation and Simulation of Fibonacci Heaps

As a practical application, an animation and simulation of Fibonacci Heaps was implemented using JEDAS. The animation dynamically visualizes the operations carried out on the data structure (Figure 3). It makes use of the original algorithm supplemented with animation instructions.

The simulation mode allows for complete interaction with the algorithm. The user can manipulate the data structure by choosing the next heap operation to be carried out. The interaction is comfortable and intuitive; the easiest way is to directly click on nodes in the animation panel and choose an option from the appearing menu. An existing heap may be saved at any time to be re-used later as an example. If a movie-mode presentation is desired, animations can also be created very easily.

In addition, a visualization of the actual and amortized time complexity is provided in a separate frame to facilitate students' understanding of this rather

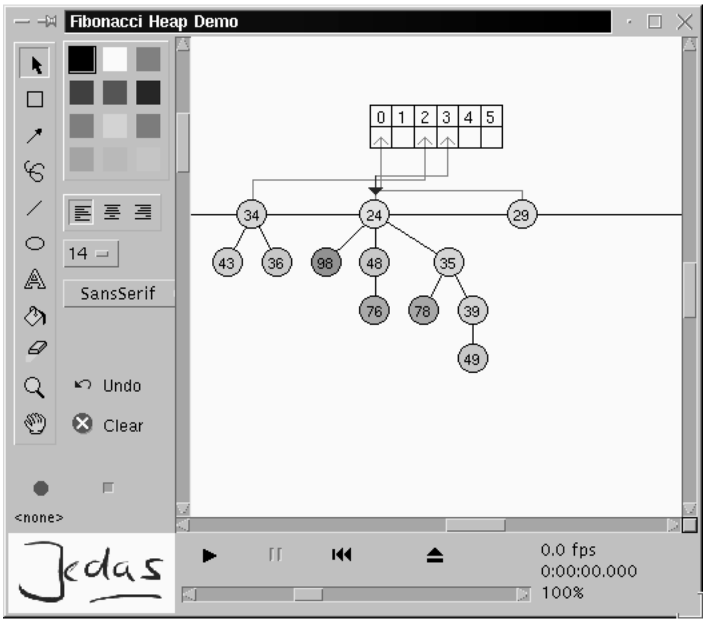

Figure 3: *JEDAS animation visualizing Fibonacci Heaps.*

complex issue. This statistics frame offers a different "view" on the same data, highlighting different aspects of the same algorithm. It was also implemented using the JEDAS library although it shows only discrete steps and no continuous animation.

One major intention was to find an implementation flexible enough to use this animation series in various application areas. As a result, it is suitable both for lecture-style presentations and less formal settings such as labs or at home.

The animation has been used in lectures on algorithms and data structures at the University of Freiburg, and a modified version animating Binomial Queues has been included in a multimedia presentation recorded with the AOF note-taking system [Müller, Ottmann 2000].

5 Annotation of MPEG Videos

As has been pointed out above, facilities for the graphical annotation of animations are much appreciated by instructors. This also holds for the case of video clips, covering animations given in a video format as well as movies which might be annotated. Similar to the options provided by a whiteboard application, simple graphical objects such as lines, squares, ellipses, as well as free-hand drawing and an online pointer are to be supported. All of the teacher's annotations made

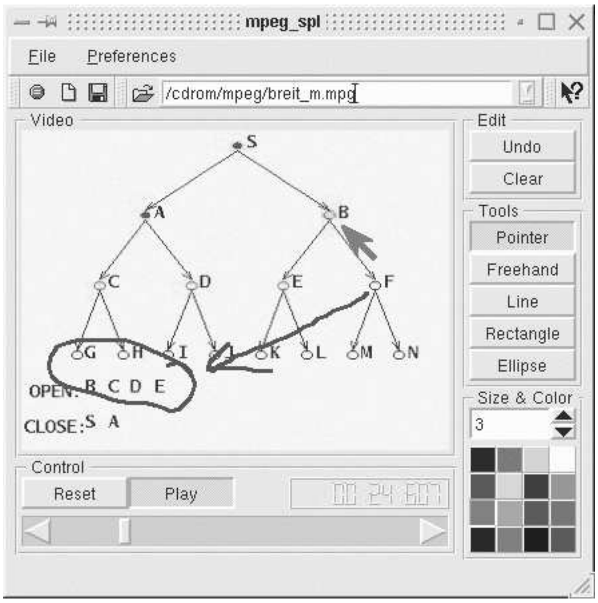

Figure 4: *Annotation of an MPEG video clip.*

in the application window during the recording of a class have to be displayed synchronously with the other data streams when playing back the video again.

It is hard to find a general solution to this problem, which would have to be independent of the video playback program that is used, in particular if the source code of that application is not available. The implementation heavily depends on the underlying window system and playing back recorded annotations could probably not be platform independent.

In the special case described here – an MPEG player developed on the basis of an existing implementation with available source code [Rowe et al. 1994] – annotation of video can be realized with less effort (see Figure 4).

After the current frame has been decoded and buffered for displaying, the annotations are added and will appear on top of the video frame. Using double buffering, the complete content of the buffer is then copied to a window visible to the viewer. The effort needed for adding the annotations is minimal compared to decoding and displaying the video itself. For storing the data of the graphical annotations, the same data structures and procedures as in the AOF system are used [see Müller 2000].

An interesting issue regarding the annotation of video (or continuous media in general) is the question of its usefulness. Animations and films are particularly doubtful in this respect, since the visibility of objects may change rapidly, making annotations in the running video rather pointless. Here it seems a better idea to occasionally pause the movie for annotating, and then continue the playback. It is, however, easy to come up with examples where graphical annotation on the running video will be helpful. Think of films for medical students showing operations. There is usually not much camera movement in such videos, which makes them suitable for annotation. The instructor can highlight certain critical areas or anticipate the next steps of an operation. A real-world example from the domain of archaeology is described in [Ottmann et al. 2000], where a camera ride through a reconstructed Roman building was shown and annotated.

Therefore, it definitely makes sense to provide options for annotation of video, both in running mode and on still pictures.

6 Case Studies in Using Animation Systems

In this section we illustrate the vast amount of possibilities of using algorithm animation beyond the movie mode by giving a few typical examples.

6.1 Inductively Inferring a Method

Instead of teaching an algorithm top-down and illustrating its behavior by running an animation on a few sample inputs, do the reverse: encourage a learner to infer the "general rule" behind a sequence of specific examples.

This works well for the classical on-line algorithms for solving the bin-packing problem. It is not difficult to infer the next-fit, first-fit, and best-fit strategies from running the (unknown) algorithms on a series of different input sequences. Making a reasonable guess in order to estimate the behavior of the different algorithms for arbitrary input sequences is, however, not that easy any more. In particular, detecting a significant difference between the first-fit and best-fit strategies requires a more thorough understanding of both algorithms.

It is obvious that the classical elementary sorting algorithms (Bubblesort, Insertion-sort, Selection-sort) and searching methods (linear, binary, exponential search) suggest similar mental experiments.

Computational geometry also offers a great number of topics where the technique of inductively inferring and formulating the method from a sequence of specific examples may be worth exploring. Think of the various strategies for exploring an unknown environment, so-called competitive on-line algorithms, and try to infer a general bound for their behavior.

6.2 Exploring the Effects of Finely Granulated Operations

Usually, the granularity of an animation is rather coarse. Think of the following example: when constructing the *minimum spanning tree* (MST) for a (planar) graph, the user may be able to specify the input interactively by selecting a set of nodes and edges, but the MST construction algorithms runs without user interaction. To use Tarjan's coloring method for constructing the MST [see Wagner 1998], two rules, the *red-coloring rule* and the *green-coloring rule*, are formulated and the construction of the MST consists in applying one of the rules as often as possible.

Instead of running this algorithm in movie mode for a given, interactively selected graph, the user should be able to interactively specify a cut in order to apply the green-coloring rule or to specify a cycle for the red-coloring rule, and then watch the effects of applying the respective rule. It is assumed that this way of using the animation system will have a much more lasting effect for understanding Tarjan's algorithm than the usual movie-mode presentation.

There are many similar occasions where a fine-granular interaction supports the understanding of algorithms. Think, for example, of the classical algorithms for rebalancing a height-balanced tree after the insertion or deletion of a node. Instead of automatically retracing the search path and carrying out a sequence of rotations, the animation system should urge the learner to specify nodes where such operations must be carried out. Experimenting that way with the animation system may thus even allow learners to discover the essence of the AVL rebalancing algorithm themselves!

Note that many animations of algorithms for restructuring graphs in general and (search) trees in particular suffer from the fact that they do not maintain *visual coherence*: pointer updates are visualized by discrete steps instead of continuous movements of lines. We think that visual coherence is an essential feature for understanding a graph-manipulating algorithm (even though this might not reflect "reality" – after all, pointer updates *are* discrete operations). Of course, a thorough cognitive study to verify or disprove our belief is yet to be done.

It is well-known that there is a close relationship between triangulations of convex polygons and full binary trees, so-called parse trees [Cormen et al. 1990]. Thus one may study the effect of an operation – such as an edge-flip – concerning the triangulation for the corresponding tree and vice versa. One may even show that any two triangulations may be transferred into each other by carrying out a sequence of rotations. Is there a correspondence between the two views? Can this be visualized appropriately and in an intuitive manner? Here we have an example where different but closely related views of structures and finely granulated operations to manipulate them seem to be crucial for understanding.

6.3 The Animation System as a Book-Keeping Tool

J. Nievergelt and his group at ETH Zürich were among the first to recognize the value of algorithm animation systems for supporting "professional" players: when using the computer to play Chess or Go, the computer may just be used as a book-keeping device to record the sequence of actions carried out by two human players. The *Smart Game Board* [Kierulf et al. 1990] is a software tool supporting this feature. Of course, one may also use the system in order to play "against" it, provided that the respective game-playing algorithms have been included in the system development.

Similarly, even systems for playing one-person games like Sokoban may be used for book-keeping: in order to study different strategies and heuristics to explore large state spaces it may be worth visualizing them and retracing the sequences of steps to invent new and more efficient search strategies. This is just one example for utilizing animation systems beyond the movie mode. The book-keeping facility, i.e., the possibility to record and replay animations, is a generic property of any algorithm animation implemented using the JEDAS library.

6.4 The Animation System as a Multimedia Authoring Tool

Producing multimedia courses is a time and cost consuming task. While the automatic recording of presentations offers an inexpensive alternative, the resulting documents often lack the professional look and didactic concepts of courses specifically developed for self-paced studying. The question is whether one can combine the two approaches, getting the advantages of both and minimizing their shortcomings.

We think of merging animation and presentation recording together in one system. The structures and methods provided by an animation system – namely the handling of moving objects – seem to be suited ideally for capturing annotations as well. Everything that is written or drawn can be treated and processed like objects of an animation. For playing back the document, the resulting stream is synchronized with all other recorded media streams.

If the system supports manipulation and re-recording of a session that is played back, one can imagine a kind of "repeated authoring" process. Errors can be deleted, new things added. Hence, the resulting documents can be improved each time they run through the cycle.

We believe that this way of multimedia authoring will be cost and time efficient while at the same time allowing the production of well-designed documents.

7 Conclusion

Today's technology, as we have seen, allows us to use animation in a vast variety of contexts. This variety will certainly grow further, both in quantity and quality. The spread of new technologies will enable more and more teachers to use animations in their classes. New authoring tools will facilitate the creation of animations for non-experts in the field. Furthermore, interactive animation systems will contribute to a more learner-centered way of teaching.

All these developments will be accompanied and supported by an increasing degree of automatization. Tools are available already which record whole presentations including animations and automatically produce multimedia documents. Visual editors are developed facilitating the authoring process of animations.

However, we also stress the need for further research in order to be able to better judge the effectiveness of animation in teaching. Some of the few studies that have been carried out in this field suggest that the learning environment plays a crucial role. We must deepen our understanding of this matter and the impact it has on our teaching if we want to use animation as a helpful tool in education.

References

1. [Brown, Sedgewick 1985] Brown, M.H.; Sedgewick, R.: "Techniques for Algorithm Animation"; IEEE Software, 2, 1 (1985), 28-39.
2. [Cormen et al. 1990] Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.: "Introduction to Algorithms"; MIT Press, Cambridge (1990).
3. [Kehoe et al. 1999] Kehoe, C.; Stasko, J.; Taylor, A.: "Rethinking the Evaluation of Algorithm Animations as Learning Aids: An Observational Study"; Technical Report, Georgia Institute of Technology (1999).
4. [Kierulf et al. 1990] Kierulf, A.; Chen, K.; Nievergelt, J.: "Smart Game Board and Go Explorer: A study in software and knowledge engineering"; J. Comm. ACM, 33, 2 (1990), 152-166.
5. [Müller 2000] Müller, R.: "Wahlfreier Zugriff in Präsentationsaufzeichnungen am Beispiel integrierter Applikationen"; Akad. Verl.-Ges. Aka, Berlin (2000).
6. [Müller, Ottmann 2000] Müller, R.; Ottmann, T.: "The 'Authoring on the Fly' System for Automated Recording and Replay of (Tele)presentations"; Special Issue on Multimedia Authoring and Presentation Techniques of ACM/Springer Multimedia Systems Journal, 8, 3 (2000).
7. [Ottmann et al. 2000] Ottmann, T.; Müller, R.; Seitz, G.; Steinert, C.: "Video in Vorlesungsaufzeichnungen mit informatikfernen Inhalten am Beispiel Archäologie"; Informatica Didactica (Zeitschrift für fachdidaktische Grundlagen der Informatik), 1, 2 (2000).
8. [Rößling et al. 2000] Rößling, G.; Schüler, M.; Freisleben, B.: "The ANIMAL Algorithm Animation Tool"; ACM 5th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2000), Helsinki, Finland. ACM Press (2000), 37-40.
9. [Rowe et al. 1994] Rowe, L.A.; Patel, K.D.; Smith, B.C.; Liu, K.: "MPEG Video in Software: Representation, Transmission, and Playback"; Proc. of High Speed Networking and Multimedia Computing, IS & T/SPIE Symp. on Elec. Imaging Sci. & Tech., San Jose (1994).
10. [Stasko 1990] Stasko, J.: "The Path-Transition Paradigm: A Practical Methodology for Adding Animation to Program Interfaces"; Journal of Visual Languages and Computing, 1, 3 (1990), 212-236.
11. [Wagner 1998] Wagner, D.: "Aufspannende Bäume minimalen Gewichts"; Ottmann, T. (ed.): Prinzipien des Algorithmenentwurfs; Spektrum Verlag, Heidelberg (1998), 173-183.