Using Java to implement a multimedia annotation environment for young children

Afrodite Sevasti
Computer Engineering and Informatics Department,
University of Patras
and
Computer Technology Institute,
61 Riga Feraiou Str., GR-262 21
Patras, Greece
(+30)-61-960316
sevastia@cti.gr

Bouras Christos
Computer Engineering and Informatics Department,
University of Patras
and
Computer Technology Institute,
61 Riga Feraiou Str., GR-262 21
Patras, Greece
+30-61-960375
bouras@cti.gr

ABSTRACT

The exceptional advent and dominance of interactive multimedia applications in our days has led to the need for their exploitation for educational, among many other, purposes. In this work, we present the design and implementation of a multimedia annotation environment for young children using the Java 2 Platform. This environment was developed to provide children of ages 4 to 8 with the opportunity to reflect upon and annotate episodes from their everyday life.

Our aim was to exploit the recent technological developments in the field of multimedia and the processing capabilities of contemporary personal computers, in order to build an annotation environment where children would be able to add multimedia annotations to videos. Apart, from the environment itself, design choices, interface realization, platform limitations and emerging solutions as well as media handling methods and performance issues are also presented.

Keywords

Interactive multimedia, video annotation, video browsing, Java, hypermedia interface, media integration and synchronization

1. INTRODUCTION

The fields of multimedia editing and annotating are currently among the rapidly evolving fields in the development of Instructional Computer Technology. Multimedia editing, to begin with, has been recognized as a widely promising tool in educational procedures. It can replace the two-dimensional written word with three-dimensional written, auditory or visual material, enriching all educational procedures and capturing the students' attention. In our days, there exist several commercial authoring applications in the field of multimedia editing and storyboarding for younger children. However, current educational software rarely offers either the child or the adult facilitator options for 'deeper interaction' and it hardly ever exploits powerful computer technologies.

In this paper we present the implementation work that took place within the framework of project "Today's Stories", part of the Long Term Research Task 4.4, (i3 –ESE, Project Nr. 29312). According to the "Today's Stories" project definition ([18]), one technological objective of the project is to develop a wearable device (actually a wearable video camera), that can capture short sequences of interest in the child's daytime. The crucial insight here is that apart from providing a fragmented history of the day of a child, recordings of an event from more than one child's perspective can be interrelated. This brings us to the second technological objective of the project, which is the development a multimedia editing tool that enables children to annotate a recorded episode with what they see, think, experience. Annotation uses expressive media, symbols (e.g., stylized faces to express various emotional states), or sound-effects (e.g., special effects to highlight for example surprise or fear). The resulting multi-medial document is to be kept as a memory and a document for future reference.

The work that took place within the scope of this second technological objective of the project, including design and implementation issues, is the subject of this paper. It must also be stated here that our work took seriously into consideration the pedagogical issues that are involved in the nature and use of such an application, as they have been approached by the "Today's Stories" consortium (see also the notion of personal autonomy as a central educational aim in [1]). After all, the project itself pays attention to social, cultural and ethical implications, as well as to the conditions for acceptance and success of deploying its technology.

For the implementation of this video exploring and annotating application, referred to from now on as the Diary

Composer (DC), the Java platform was elected. More specifically, the Java 2 SDK, Standard Edition, v 1.2.2, developed by Sun Microsystems, Inc. and the Java Media Framework (JMF) 2.0 API developed by Sun Microsystems, Inc. and IBM were used. This choice was made for several reasons, with the first one being to assure complete platform independence and that the final version of the application could also be accessible through the Internet, for future use from the children's home. The fact that Java programs are compiled for the Java Virtual Machine (JVM) enables Java programs to be executed on a variety of platforms, provided that the JVM is implemented for each one of these platforms [19].

Apart from that, the release of version 2.0 of the Java Media Framework (JMF) with a much richer set of features as well as the advanced features provided by the Swing 1.1.1 API (included in the Java Foundation Classes (JFC) API of Java 2 SDK, Standard Edition, v 1.2.2) made the Java 2 Platform appropriate for our implementation.

In this paper, we initially make a thorough presentation of the standardization, research and implementation attempts in the field of multimedia annotation. Consequently, we are presenting an overview of the DC design and architecture, describing briefly the functionality provided by the application. The remainder of the paper analyses the implementation techniques that were used for implementing that functionality as well as the limitations that were imposed by the platform used for the implementation, accompanied by performance issues. Finally, we are describing our future work on this application.

2. RELATED WORK

A worthwhile approach to the standardization of multimedia annotating is the work of the EBU/SMPTE Task Force for Harmonized Standards for the Exchange of Program Material as Bit streams. The Task Force has produced [7], which in section 4, 'Wrappers and Metadata' gives a thorough terminology and structure (close to that defined by the Digital Audio-Video Council (DAVIC)) for how physical media (video, audio, data of various kinds including captions, graphics, still images, text etc.) can be linked together, for streaming of program material and stored in file systems and on servers.

Another attempt towards the direction of standards for multimedia annotation was made by the Workshop on MMI (Metadata for Multimedia Information) conducted by the European Committee for Standardisation and Information Society Standardisation System (CEN/ISSS) from February 1998 until June 1999. The workshop resulted into deliverables on the requirements and model for metadata for multimedia information.

MPEG-7 is an ISO/IEC standard being developed by MPEG (Moving Picture Experts Group), formally named "Multimedia Content Description Interface". It aims to create a standard for describing the multimedia content data that will support some degree of interpretation of the information's meaning, which can be passed onto, or accessed by, a device or a computer code. In [11] it is stated that MPEG-7 *must* accommodate audio-visual material and take advantage of the ability to associate descriptive information within video streams at various stages of video production. As an example for this, information captured or annotated during shooting, and post-production edit lists are suggested. Based on these principles among others, MPEG-7 will work on making a global standardisation for multimedia annotations.

Apart from standardisation efforts, a lot of research work is performed in the field of video editing and multimedia annotation. An interesting work that deals mainly with non-linear video navigation and organization is presented in [9]. The author introduces the notion of 'hypermovies': hyper-documents that only consist of movie nodes. These nodes are entities comprising not only of the video as content but also of additional cinematic information which is synchronized with the video and can be made visible on demand to support navigation. The MovieCatalog tool presented, uses the Apple Macintosh Finder look-and-feel in order to organize movie segments, by displaying on the screen one thumbnail for each one of them, while the Segmenter tool uses the multiple tracks that the Quicktime format supports, to add additional cinematic information to different tracks.

The multiple tracks provided by the Quicktime format, are also utilized for the development of a Movie Authoring and Design system called 'MAD' which is presented in [8]. MAD facilitates the process of creating dynamic visual presentations by simultaneously allowing easy structure creation or modification of motion pictures and visualization of the result of those modifications. The principles behind MAD include hierarchical multimedia document representation, the flexible inclusion and combination of words, images, sounds, and video sequences, and real-time playback of a rough version of the final film at any time in the process.

An effort to reinforce a document design methodology to a hypermedia document is presented in [15]. Here, the authors point out that hypermedia applications are real-time, dynamic and depend on user interactions and therefore authoring techniques usually postpone design validation to the run-time stage. Moreover, they emphasize the fact that hypermedia documents can contain inconsistencies, which are stemming from the temporal constraints that are applied to their components through various relationships among anchors. The document design methodology proposed, translates a high-level model of a hypermedia document into an RT-LOTOS formal specification, on which standard reachability analysis may be applied for verification purposes.

Another interesting work, the results of which were seriously taken into account while developing our DC application, is presented in [10]. Here, the authors investigate the users' assimilation and understanding of the informational content of multimedia clips, to conclude upon a significant result: The quality of video clips can be severely degraded without the user having to perceive any significant loss of informational content.

The authors of [14], developed 'CueVideo' in order to provide a solution to effective video cataloging and browsing. The first phase of the 'CueVideo' system involves the so-called integrated cataloging which includes segmenting the video files into shots and adding image, text, speech etc. as annotations. In this way, the user can incorporate the type of semantic information that automatic techniques would fail to obtain and which facilitates content characterization, browsing and retrieval. An interactive video authoring system that supports the video object annotation capability is presented in [6]. The so-called 'Zodiac' system allows users to associate annotations, such as text, image and audio, to moving objects in a video sequence. This work makes a

step even further on the subject of video annotation, since it doesn't annotate video frames but objects in video frames.

The approach followed in the ACTS project no. AC082, called DIANE (Design, Implementation and Operation of a Distributed Annotation Environment) ([2],[3]), was to allow the recording if an arbitrary application output as the basic content of a multimedia document and to annotate it with all kind of media available to the user. This was achieved by the concept of a multimedia annotated document consisting of two distinct parts: recorded application output and annotations given by a user in various media including text, audio, video and pointer movements. By providing generic techniques to record output of arbitrary applications, DIANE implemented an annotations' recording tool independent of any application context.

A brief but thorough presentation of commercial authoring applications in the fields of multimedia editing for young children is given in [4]. These applications comprise worthwhile solutions to the direction of offering children the opportunity to experiment and create with the use of multimedia. However, none of them offers the functionality of adding annotations to videos and most of them include textual components for user interaction – something that we definitely wanted to avoid in our case.

A conclusion that can easily be drawn by this extensive research in the standardisation and implementation initiatives on annotation of multimedia is that the corresponding technologies have not matured yet. Standardisation efforts, with that of the MPEG-7 standard being the most significant, are still under development. The MPEG-7 standard for example is due to no sooner than 2001. Implementation efforts, on the other hand, tend to use their own method of annotating multimedia, making their solutions proprietary. In this work, we present our implementation solution for adding multimedia annotations to videos and we describe what our future work will focus on, in order to conform to the upcoming standards.

3. APPLICATION OVERVIEW

The DC application was designed so that it provides two distinct components and corresponding interfaces: one for making the video recordings accessible to the DC users, allowing navigation and selection of the video/videos to be annotated and one for providing the tools and infrastructure for annotation. We call the first component of the application the 'Video Explorer' and the second one the 'Annotation Panel'.

3.1 The Video Explorer

This component of the application was designed to provide the functionality of displaying a representative frame (actually the first frame) from each video recording stored into the DC system. These frames are called 'thumbnails' and it was a challenge of the design and implementation phase to organize and display them on screen in an efficient manner that would also be comprehensible to children of ages 4 to 8.

According to [20], all authoring interfaces use authoring metaphors for their implementation such as a slide-show metaphor, a book metaphor, a timeline metaphor, an icon metaphor etc. For the DC implementation, the set of videos recorded needed to be somehow structured and included in a multimedia database. The "Today's Stories" objectives indicated that the storage and retrieval should avoid using textual categories.

For the implementation of the Video Explorer, the timeline metaphor was adopted. Generally speaking, a timeline metaphor is an authoring metaphor where objects and events are placed on the time scale in correct relationship. For the case of the Video Explorer, video thumbnails are organized in groups according to the identity of their owner, in other words, according to which child shot them. Each group of video thumbnails is then distributed along a timeline, which in interface terms is represented by a straight line extending from one point of real-world time to another. The distribution of video thumbnails along the timeline is proportional to the shooting time of each video recording.

Another significant component of the Video Explorer is the help feature, envisaged by the 'Genie' character. This 'Genie' has been implemented to have drag-and-drop features within the Video Explorer (it also appears in the Annotation Panel) environment. Dropping the 'Genie' on a certain visual component of the Video Explorer initiates an event that provides the DC user with an audio description and a brief listing of the visual component's functionality. This implementation of the help feature was based on the JFC/Swing Containment Model (see [12],[13]) which allows for placing interface components in different layers within the same container. More details on this will be provided in the 'Implementation Issues' further down.

Figure 1. The Video Explorer layout

The Video Explorer supports simultaneous use of more than one users, by presenting more than one timelines in cases when more than one children share the same workspace (see Figure 1). During the videos' storing procedure, metadata that indicate which of the videos are shootings of the same incident by different children, are also stored in the system. During the implementation of the DC, several methods were developed that associate related video files and their visual representations on the Video Explorer interface. Thus, video files of the same incident shot from different perspectives, form a so-called 'hyper-video' in the DC. This hyper-video, containing one, two or three separate video files is the entity handled by the annotation methods of our DC application. It is actually treated as an autonomous unit (for annotation, auto-save etc. procedures) to which all annotations are added, from the moment it is realized.

The functionality supported by the Video Explorer component includes the 'delete' operation. This operation has been implemented in such a way that all annotations of an annotated video are erased when the video thumbnail is dragged and dropped upon the 'Dustbin' visual component of the interface. Future versions of the DC will also support navigating through videos from different dates, by the use of the 'Diary' visual metaphor (see Figure 1)

Each single video thumbnail or hyper-video comprises a 'hyperlink'. This 'hyperlink', when clicked or pressed upon (this depends on whether a mouse or a touch-screen is being used for interface I/O) initializes an annotation session, by opening the second component of the DC application, the so-called Annotation Panel.

3.2 The Annotation Panel

The second component of the DC has as its main functionality to provide the user with the tools to add image and sound annotations to the video/videos already selected via the Video Explorer.

The dominant still controversial features of the Annotation Panel implementation are its dynamic nature and simplicity. Bearing in mind that the tool aims at very young children and that the procedure of annotation actually can be broken into two phases (adding/removing annotations and playback of the video in its current state, containing all annotations previously added), appropriate methods had to be implemented. To be more specific an annotation procedure consists of the following steps:

- Start the video playback
- Pause playback
- Either drop one or more annotations to the video's visual component (add) or drag previously added annotations from the videos visual component (remove)
- Resume playback

The last three steps are iterated as many times as the user wishes to.

A playback procedure consists of the following steps:

- Rewind the video up to its first frame
- Start the video playback
- Pause and resume playback

Again, these steps can be iterated as many times as the user wishes to.

The design choice made was to merge the annotating and playback procedures so that the two distinct phases would become transparent to the users of the DC. The DC application allows for steps of the annotation procedure to be interrupted for initializing a playback procedure and vice versa. In other words, the user is provided with the ability to interrupt the annotation procedure, in order to go back and find out how he has done so far, but during playback he can still pause the video reproduction and add annotations. We have called this 'dynamic annotation' and more implementation details will be given in the 'Implementation Issues' section.

Apart from the 'Genie' character that implements the help feature in the Annotation Panel in an identical to that in the Video Explorer way, the Annotation Panel provides the user with a totally different set of functionality. Three simple video control buttons are provided to the user for controlling playback of the hyper-video being annotated. According to the Video Explorer implementation, more than one videos might be chosen for simultaneous annotation, in cases more than one video recordings of the same incident, forming a hyper-video, exist. However, this hyper-video depicts the same incident from two or three different perspectives and there was no point in providing the users of the DC with the possibility to start/stop/resume each video component of the hyper-video separately. Therefore, the video control buttons are placed on one control panel, which is common to all the videos that are being annotated at the moment. From the implementation point of view, this approach required video synchronization techniques, which will be analyzed further in the 'Implementation Issues' section, to be adopted.

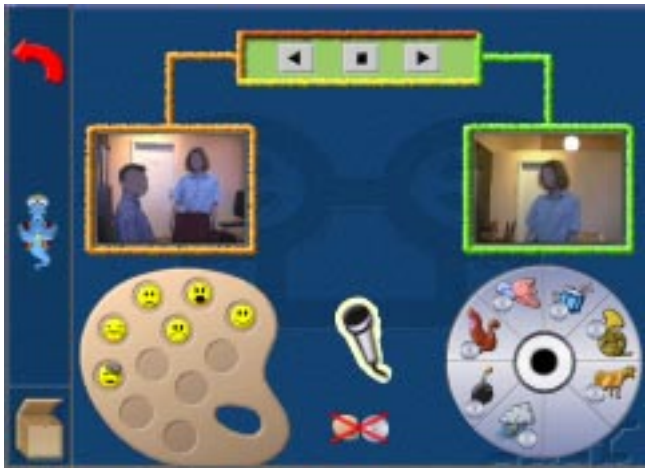

Figure 2. The Annotation Panel layout

For the current version of the DC, two types of annotation palettes were implemented: one for image annotations and one for sound annotations (represented by images as well for the purposes of the DC interface). The first palette (see bottom left of Figure 2), contains image annotations that are depicting emotions while the second one (see bottom right of Figure 2) contains image annotations that are depicting sounds but also represent sounds. The user is provided with the functionality of pausing the video at any moment and adding to it an unlimited number of image and sound annotations, in order to comment the incident that has been recorded.

Both image and sound annotations are implemented in such a way that present the users with a 'sticky' behavior: annotations can be dragged towards any direction within the DC interface but can be dropped only on one of the video frames. Dropping an annotation outside a video frame, results in the annotation moving back to its original position in the palette. The latter provides the users with the ability to remove annotations from a video, simply by dragging away any annotation that has been 'stuck' on the video and dropping it outside the video frame. Adding a sound annotation to a video frame does not only result into the correspondent icon being added to the video as an image annotation: it also 'adds' the sound associated with it to the sound track of the video, so that the sound annotation is reproduced during playback, at the time it was placed during annotation.

Apart from the predefined set of sound annotations, new sound annotations (e.g. spoken words) can also be inserted to the current hyper-video. The user has to pause video playback at the desired video frame and speak to his workstation's microphone while pressing the microphone icon between the two palettes (see Figure 2). In this way, all kind of sound annotations can be 'added' to the video's audio track.

Except for removing annotations one by one, the Annotation Panel provides the user with the opportunity to reset the hyper-video being currently annotated, thus removing all annotations at once (by pressing the small icon under the microphone in Figure 2). Finally, there is a 'Back' button that can be used from users to finish their annotating session and return to the Video Explorer either for further navigation through videos or for choosing another hyper-video to annotate.

4. IMPLEMENTATION ISSUES

4.1 Annotation

For the implementation of the annotation functionality in our DC application, we co-estimated the related work presented in section 2 of this paper and the requirements specified by the "Today's Stories" consortium as far the multimedia annotating application to be developed was concerned.

Due to the lack of mature standards and in order to achieve acceptable performance and platform independence in application execution and video supported formats, we designed a proprietary still open and configurable annotation system. The main idea was to create transparent to the end user data structures that would hold hyper-videos and all kinds of multimedia annotations attached to them in an efficient and consistent way. This way, we bypassed the limitations and performance issues rising from all our attempts to encode video and annotations together.

All of our annotation methods follow the principle of associating an annotation with the frame of the video to which the annotation was added. In this way, predefined images and sounds as well as recorded sounds are associated in dynamic data structures with the video frame being annotated by them. During the application runtime, this association is dynamic and configurable. This approach makes the feature of 'dynamic annotation' described in section 3.2 feasible.

The annotation procedure can be interrupted by the user at any time in order for him to be able to preview his annotations so far. The application then reads from those dynamic data structures in order to represent the annotated hyper-video playback to the user. This is achieved by displaying images and opening sound players at those positions during playback, where the data structures' contents appoint. At any moment, the user can remove any annotation he wishes. This functionality is internally implemented by removing the correspondent entries from these data structures and re-ordering the data structures' contents.

All these dynamic interactions are possible during an annotation session. As soon as the user wishes to interrupt the annotating procedure, annotation data together with the hyper-videos they refer to, are stored permanently to the system for future use. For the current version of the application, we have implemented a proprietary scheme for storing all this information.

The implementation approach described here has the following advantages:

- Supports a wide variety of video formats for the video files that are annotated within the application (in [16], a list of the supported by the JMF API media formats is provided)
- An unlimited set of different kinds and formats of multimedia data can be used as annotations (images, sounds etc.)
- Performance is preserved in satisfactory levels independently of the amount/type of annotations inserted, due to the fact that annotations are stored separately from the raw video data and not encoded within it.

The main disadvantage of our implementation approach is the fact that annotated hyper-videos are stored in a proprietary format, readable by our application alone. In section 5, we describe how our future work will proceed in order to eliminate this disadvantage.

4.2 User Interface

For the purposes of the annotation functionality, the implementation procedure exploited the potentials of the JFC/Swing Lightweight Component Framework and Containment Model ([12],[13]) so that the DC application was developed on a multi-containment, multi-layer infrastructure. The JFC/Swing Lightweight Component Framework and Containment Model were two of the major enhancements of Swing over the Abstract Window Toolkit (AWT) that the Java platform used to provide for User Interface (UI) development. Actually, the DC application would be impossible to implement without these two features of Swing.

The Lightweight Component Framework facilitated the implementation procedure in the sense that the application was organized in many different components, the most complicated of which contained the simpler ones, thus acting as containers. All methods implementing objects' behavior within the application itself and the application UI, were accessible by the appropriate levels of containment hierarchy in order to provide a consistent and reliable look and feel throughout the application. An indicative diagram that depicts a large proportion of the containment hierarchy of the DC implementation objects is shown in Figure 3.

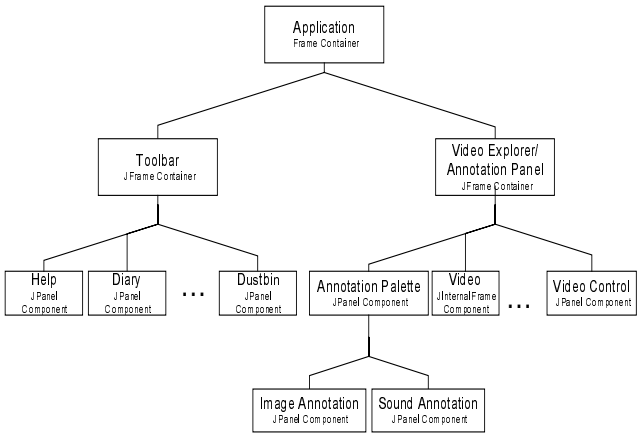

Figure 3. The multi-containment infrastructure

Root containers of the containment hierarchy have been implemented in such a way that act as mediators among their components. They listen for events from certain components (e.g.

a mouse 'click' event on a video control object) and generate other events according to the application's functionality protocol (e.g. starting, stopping or rewinding of the initiated video player/s depending on which video control object was triggered). In this way, leaf components of the containment hierarchy do not communicate directly with each other. Instead, each event climbs up in the hierarchy tree so that all listener objects are informed about it in order for the appropriate methods to be called.

The JFC/Swing Containment Model allows for organizing components in different layers within their container. Actually, this feature is provided by the group of the heavyweight container classes in Swing and is referred to as the nested-container hierarchy in [13]. According to this infrastructure, the Annotation Panel container was organized in layers: the background component controls and coordinates all other layers. Videos and video controls are placed two layers above on the so-called Modal layer while annotations are placed on the topmost Drag layer (see Figure 4).

This UI architecture was adopted for several reasons and resolved most of the implementation problems. One of these reasons was to make the heavyweight visual component of each video player cooperate with the rest of the application, being placed over the UI background and under the annotations' layer at the same time. Of course, this architecture facilitated the implementation of the drag and drop behavior of annotations and the 'Genie' object in the best possible way. Annotation and 'Genie' objects' behavior is defined by a set of methods that allow them to move on the UI drag layer, thus creating the impression that they 'float' over all other UI components. Their drag-and-drop behavior was implemented over the mouse listening interface that the Java 2 SDK provides.

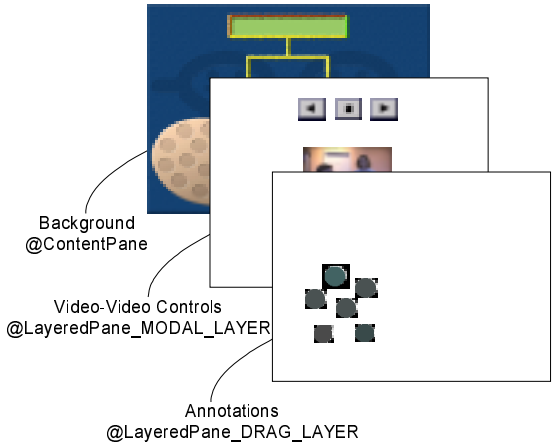

Figure 4. The multi-layer infrastructure

4.3 Media Handling

The release of version 2.0 of the Java Media Framework (JMF) API from Sun Microsystems, Inc. and IBM has undoubtedly provided Java platform programmers with a much wider set of features for inserting and handling multimedia in their applications. The major challenges that the DC implementation had to face was to ensure the best quality possible while loading more than one video/audio players, synchronize these players and monitor their behavior throughout the application runtime. In this section, we present how the JMF API was used for the DC implementation and the design choices made in order to achieve the desired functionality.

According to [17], the JMF specification defines APIs for displaying time-based media. JMF players share a common model for timekeeping and synchronization and JMF clocks define the basic timing and synchronization operations. Also, according to [5], JMF does not build the functionality of constant media progression tracking into each media player. Based on the above, we had to implement several mechanisms for accessing video data by frame number instead of media time, for keeping track of the video data progression in frame numbers and for generating events according to the current video frame number.

One of the dominant components for video/audio management within the DC was the Frame Positioning Control (FPC) interface of the JMF API ([5],[16]). This interface was used for accessing the individual frames of each video file, a feature which is not built-in within JMF. This limitation produced the first obstacles in the Video Explorer implementation, where a player interface and consecutively an FPC interface for each video had to be initialized in order to access the first representative frame of each video and transform it to an image thumbnail. This procedure was very consuming in terms of resources in cases such as the one depicted in Figure 1, where ten different video thumbnails were to be created.

The design choice that was made here deviated this problem by removing from the DC implementation the processing burden of extracting a thumbnail from each video file. Instead, an asynchronous application that uses the FPC interface to produce a thumbnail, runs every time a new video file is stored in the DC infrastructure. This standalone application is transparent to the user and produces one image file for each video file, to be used as the video thumbnail in the Video Explorer.

In order for the annotation feature of the DC to be implemented, we had to keep track of the video progression in frame numbers or, in other words, be aware of the number of the current video frame being displayed on screen, during the annotation/playback procedure. Whenever video playback is paused and annotations are added to or removed from it, the current frame number is used to associate annotations with the video, as it has already been explained in section 4.1.

However, keeping track of the video progression in terms of frame number, required the combination of the FPC interface and its mapTimeToFrame method. Actually, calling this method on the FPC interface of a video player is the only way to monitor a video player's progression in frame numbers within JMF. This procedure is often performed during the DC runtime and more specifically every time the application needs to be informed about the exact video frame number to which one or more annotations have been added. This fact could not be ignored by our implementation and it is one of the major performance drawbacks of the DC application.

Things become even more complicated if we attempt to introduce the functionality of displaying annotations previously added to a video file, during playback. This functionality requires constant monitoring of the video player's current frame number, so that an annotation display event is initiated when the video playback approaches the video frame number where the annotation was added. For this functionality and since JMF players cannot

produce such events themselves, we had to implement a thread running in parallel to the video player/s.

This thread has as its main duty to monitor the player status, and place annotations to their position inside a video's visual component, according to the data recorded during the annotation procedure. It receives as input the content of the data structures, where data, associating video frames with annotations, are stored. We will refer to these data as annotation data entries (ADE) from now on. Each ADE, consists of an annotation's insertion frame number (AIFN) and an annotation identifier (AI). In fact, the thread implemented complies with the following algorithm:

Thread activates itself only when video player is in "Started" state

Thread polls the video player for the current video frame number (CVFN)

Thread compares CVFN with the subset of ADEs for which AIFN<CVFN

WHILE (AIFN<CVFN+15 and AIFN>CVFN-15)

> *Add the annotation referred to by this ADE's AI to the video's visual component*

This algorithm ensures that although annotations are actually added to one frame of the videos (AIFN), they are displayed for a window of 30 video frames (starting from 15 frames before the AIFN and finishing at 15 frames after the AIFN) so that the user can perceive their existence. The main drawback of this approach has turned out to be the burden placed upon the application's performance by the thread introduced.

The implementation of the annotation functionality within the DC environment includes also synchronization techniques for the cases when two or three videos forming a hyper-video are being simultaneously annotated. Generally speaking, JMF provides the functionality of synchronization of multiple players in such a way that the programmer has to define a master player, the controls of which are responsible for all other players stated as 'slaves'. For the DC purposes we had to implement methods that designated as master player in a group of two or three videos the one with the maximum duration. We also made several experiments that resulted into the conclusion that equal frame rates among the members of a video group are ensuring best synchronization and performance behavior.

Finally, the JMF API does not provide a reliable mechanism for interfering with the video players' playback rate. In fact, not all players are guaranteed to allow their playback rate to be adjusted, that is reduced or increased. This fact prevented us from being able to provide videos' fast forward and rewind functionality within our application in its current version. This issue is part of our future work and might also be resolved in future versions of the JMF API.

4.4 Performance Issues-Memory management

A lot of performance issues occurred during the implementation procedure of the DC application, some of which have already been mentioned in previous sections of this paper.

A quantative amount of experimenting took place in order to determine the appropriate video format characteristics that would allow qualitative simultaneous playback of three different video files in the Annotation Panel component. Results designated as the only solution recommended for good performance that of reducing the video files' frame rate to numbers less than or equal to 15 frames per second. According to the results of [10], we are allowed to do so for the sake of playback quality, without putting in stake significant loss of informational content.

Another significant issue related to performance, that had to be dealt with, was that of memory management. The nature of the DC application requires for tentative memory interactions and extensive care had to be taken for the introduction of efficient memory management into the application. Keeping in mind that Java does not provide sophisticated and automated mechanisms for memory management and garbage disposal, several methods were implemented for disposal of the extensive memory resources occupied by video players. These methods anticipate for master-slave relationships between video players and clean up most of the system's memory resources each time an annotation session is terminated.

5. FUTURE WORK-CONCLUSIONS

The DC application presented in this paper comprises the first fully functional version of a tool that will be further enhanced with functionality and improved. This will be done according to the feedback that will be provided from extended trials in school environments within the "Today's Stories" project time plan and the pedagogical analysis of the application's use and nature.

From the implementation point of view, functionality to be added includes:

- To make the DC environment accessible over the Internet
- An interface for the insertion of custom image annotations to the application
- A hierarchical structure that will support multiple annotation palettes, categorized according to different thematic categories
- A navigator-browser for the users' video recordings from previous dates
- More sophisticated video controls, such as fast forward and rewind

Apart from adding functionality to the existent tool, our future work will also investigate the possibilities to introduce multimedia annotations in widely accepted video content types such as QuickTime. QuickTime architecture and file format offer themselves for annotation insertion, since they organize media data in synchronized tracks. However, there are still limitations to the data types that can be used as annotations and of course to the platforms over which an annotated QuickTime movie can be presented. We are currently working with the QuickTime for Java API in order to make QuickTime and its annotation techniques accessible to all the Java compliant platforms.

Generally speaking, we can conclude that our work and all other efforts should move towards the direction of multimedia annotations' standardization. It is for sure that the upcoming MPEG-7 standard will contribute significantly towards this direction.

6. ACKNOWLEDGMENTS

We would like to thank all our partners in the "Today's Stories" consortium for their feedback and collaboration in the procedure of defining the functional specifications of the DC application.

We would also like to mention that the application's interface graphics and layout that appear in Figure 1 and Figure 2 were designed by the Assistive Technology & Human-Computer Laboratory, Institute of Computer Science, Foundation for Research & Technology-FORTH, Crete, Greece. The videos, the thumbnails of which appear in Figure 1 and Figure 2, were produced by Marilyn Panayi and David Roy and were shot in Denmark, at the KISS – NIS Laboratory and Nr. Brody Skole.

7. REFERENCES

[1] Aviram, A., *Personal Autonomy And The Flexible School*, International Review of Education 39(5), Kluwer Academic Publishers, printed in the Netherlands, 1993, 419-433

[2] Benz, H., Bessler, S., Fischer, S., Hager, M., and Mecklenburg, R. DIANE: "A Multimedia Annotation System" in *Proceedings of the Second European Conference on Multimedia Applications, Services and Techniques (ECMAST'97)* (Milan IT, May 1997), Fdida, S., and Morganti, M. (Eds.), Lecture Notes in Computer Science 1242, Springer Verlag, Berlin, 183-198.

[3] Benz, H., Fischer, S., Mecklenburg, R., and Dermler, G. DIANE - "Hypermedia Documents in a Distributed Annotation Environment" in *Proceedings of the Conference on Hypertext - Information Retrieval - Multimedia (HIM'97)* (Dortmund DE, September 1997), Norbert, F., Gisbert, D., and Tochtermann, K.(Eds.), Schriften zur Informatik, UVK Universitaetsverlag, Konstanz, 293-306.

[4] Bouras, C., Kapoulas, V., Konidaris, A., Ramahlo, M., Sevasti, A., and Van de Velde, W. "Diary Composer: Supporting Reflection on Past Events for Young Children" To appear at the *ED-MEDIA 2000-World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Montréal (Canada), June 26-July 1, 2000.

[5] Carmo, L.de. *Core Java Media Framework*, Prentice Hall PTR, Upper Saddle River NJ, 1st edition (June 24, 1999).

[6] Chiueh, T., Mitra, T., Neogi, A., and Yang, C.K. Zodiac: A "History Interactive Video Authoring System" in *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98)* (Bristol UK, September 12-16, 1998), 435-443.

[7] EBU/SMPTE Task Force for Harmonized Standards for the Exchange of Programme Material as Bitstreams. Final Report: Analyses and Results, August 1998.

[8] Baecker, R., Rosenthal, A.J., Friedlander, N., Smith, E., and Cohen, A. "A Multimedia System for Authoring Motion Pictures" in *Proceedings of the 4th ACM International Multimedia Conference (Multimedia'96)* (Boston MA, November 18-22, 1996), 31-42

[9] Geissler, J. "Surfing the Movie Space: Advanced Navigation in Movie--Only Hypermedia" in *Proceedings of the 3rd ACM International Multimedia Conference (Multimedia'95)* (San Francisco CA, November 5-9, 1995), 391-400.

[10] Ghinea, G., and Thomas, J.P. "QoS Impact on User Perception and Understanding of Multimedia Video Clips" in *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98)* (Bristol UK, September 12-16, 1998), 49-54.

[11] International Organisation For Standardisation (ISO/IEC JTC1/SC29/WG11) - Requirements Group (Adam Lindsay, Editor), MPEG-7 Applications Document v.9 (ISO/IEC JTC1/SC29/WG11/N2861), (Vancouver CA, July 1999).

[12] Pantham, S. *Pure JFC Swing*, Sams Publishing, Indianapolis IN, 1999.

[13] Piroumian, V. *Java Gui Development: The Authoritative Solution*, Sams Publishing, Indianapolis IN, 1999.

[14] Ponceleon, D., Srinivasan, S., Amir, A., Petkovic, D., and Diklic, D. "Key to Effective Video Retrieval: Effective Cataloging and Browsing" in *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98)* (Bristol UK, September 12-16, 1998), 99-107.

[15] Santos, C.A.S., Soares, L.F.G., de Souza, G.L., and Courtiat, J.-P. "Design Methodology for Formal Validation of Hypermedia Documents" in *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98)* (Bristol UK, September 12-16, 1998), 39-48.

[16] Sun Microsystems Inc. *Java Media Framework API Guide*, September 3, 1999, v. 0.8.

[17] Sun Microsystems Inc., Silicon Graphics Inc., and Intel Corporation. *Java Media Players*, Version 1.0.5., May 11, 1998.

[18] Today's Stories: i^3 –ESE (Long Term Research Task 4.4) Project Nr. 29312, Project Web Site found at: http://stories.starlab.org/about.htm

[19] Weber, J.L. *Special Edition Using Java 2 Platform*, Que Publishing, 1999.

[20] Yoo, S. Multimedia Authoring/ Scripting. Course seminar. Oct. 14, 1995. Found at: http://mmlab.snu.ac.kr/course/mmseminar/temp/YsPres.html