Jl. of Educational Multimedia and Hypermedia (2002) **11**(4), 363-385

An Environment for the Annotation of Multimedia Material by Younger Children

CHRISTOS BOURAS, APOSTOLOS GKAMAS,
AND AFRODITE SEVASTI
Research Academic Computer Technology Institute and University of Patras
Greece
bouras@cti.gr
gkamas@cti.gr
sevastia@cti.gr

The exceptional advent of interactive multimedia applications has led to the need for their exploitation for educational purposes. In this article, the design and implementation of a multimedia annotation environment for young children is presented. This environment was developed to provide children aged 4 - 8 with the opportunity to reflect upon and annotate episodes from their everyday life. The aim was to exploit the latest technological developments in the field of multimedia, 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, as well as media handling methods and performance issues are also presented.

The fields of multimedia editing and annotating are 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. Several commercial authoring applications in the field of multimedia editing and storyboarding for younger children exist. However, 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 article 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) is presented. According to the “Today’s Stories” project definition (Today’s Stories, 2001), one technological objective of the project is the development a multimedia editing tool that enables children to annotate a recorded episode with what they see, think, and experience. Annotation uses expressive media, symbols (e.g., stylised 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. This work also took 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 (Aviram, 1993)).

For the implementation of this video exploring and annotating application, referred to from now on as the Diary Composer (DC), the Java platform was selected. 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.

Concerning efforts related to our 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 (EBU/SMPTE, 1998) gives a thorough terminology and structure for how physical media (video, audio, data of various kinds including captions, graphics, still images, text, etc.) can be linked together, for streaming 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 in deliverables on the requirements and a model for metadata for multimedia information.

MPEG-7 is an ISO/IEC standard being developed by Moving Picture Experts Group (MPEG), 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 Ghinea & Thomas (1998), 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. 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 is performed in the field of video editing and multimedia annotation. An interesting work that dealt mainly with nonlinear video navigation and organization is presented in Geissler (1995). The author introduced the notion of "hypermovies": hyper-documents that only consist of movie nodes. These nodes are entities comprised of the video as content, and also additional cinematic information, which is synchronized with the video and can be made visible on demand to support navigation.

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 Baecker, Rosenthal, Friedlander, Smith, and Cohen (1996). 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.

An effort to reinforce a document design methodology to a hypermedia document is presented in (Santos, Soares, deSouza, & Courtiat, 1998). Here, the authors pointed out that hypermedia applications are real-time, dynamic, and depend on user interactions. 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.

Another interesting work, the results of which were seriously taken into account while developing our DC application, is presented in Ghinea and Thomas (1998). Here, the authors investigated the users' assimilation and understanding of the informational content of multimedia clips, to conclude 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 Ponceleon, Sriniyasan, Amir, Petkovic, and Diklic (1998), developed "CueVideo" to provide a solution to effective video cataloguing and browsing. The first phase of the "CueVideo" system involves the so-called integrated cataloguing, which includes segmenting the video files into shots and adding image, text, speech, and so forth as annotations. An interactive video authoring system that supports the video object annotation capability is presented in Chiueh, Mitra, Neogi, and Yang (1998). The so-called "Zodiac" system allows users to associate annotations, such as text, image, and audio, to move objects in a video sequence.

The approach followed in the ACTS project no. AC082, called DIANE (Design, Implementation and Operation of a Distributed Annotation Environment) (Benz, Bessler, Fischer, Hager, & Mecklenburg, 1997a; Benz, Fischer, Mecklenburg, & Dermler, 1997b), was to allow the recording of an arbitrary application output as the basic content of a multimedia document and to annotate it with all kinds of media available to the user. Finally, a brief but thorough presentation of commercial authoring applications in the fields of multimedia editing for young children is given in Bouras et al. (2000).

In this article, after this presentation of the standardization, research, and implementation attempts in the field of multimedia annotation, an overview of the DC design and architecture is presented, briefly describing the functionality provided by the application. The remainder of the article 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, the future work on this application is described.

APPLICATION OVERVIEW

The Diary Composer

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."

The Video Explorer

The Video Explorer was designed to provide functionality for browsing through previously shot videos to as many as three different users/children simultaneously. The best way to represent a video file (presupposing that file naming is prohibited due to the fact that written word should be avoided), is by using a still image from the videos contents: an image that will depict one of the most characteristic events recorded in the video file. For the purposes of the DC application, the display of a representative frame (actually the first frame) from each video recording stored in the DC system, was adopted. These frames are called "thumbnails" and are stored as image files, separate from the corresponding video files in the system.

Once the thumbnails are created, they have to be appropriately displayed on the workstation's screen so that the Video Explorer will act as an efficient browser of the video archives for its users. According to Yoo (1995), all authoring interfaces use authoring metaphors for their implementation such as a slide-show metaphor, a book metaphor, a timeline metaphor, an icon metaphor, and so forth. For the implementation of the Video Explorer, the timeline metaphor was adopted. 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. For videos that have already been annotated, small icons on the bottom of the corresponding thumbnails depict their annotations' nature (see the video on the upper timeline of Figure 1).

The "Today's Stories" system infrastructure anticipates for automatic authentication of the users as soon as they approach the DC system (by the use of infra-red beacons and receivers), however it was necessary to implement a mechanism for the users' orientation within the Video Explorer interface. This mechanism allows the DC user to define a digitised photograph or image as his representative within the DC user interface and a colour to identify his material (Figure 1).

The Video Explorer supports simultaneous use of more than one user, by presenting more than one timeline in cases where more than one child shares the same workspace (Figure 1). During the videos' storing procedure, meta-data that indicate which of the videos are shootings of the same incident by different children, are also stored in the system.

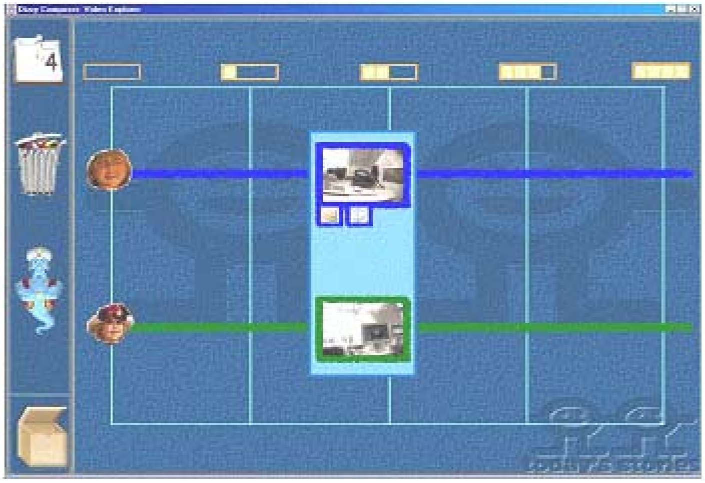

Figure 1. The video explorer layout

In a group of two or three interrelated videos of the same incident, one is identified as “parent,” and all of them are related in a precise manner. In Figure 1, the two video thumbnails from the two timelines consist of recordings of the same incident from different perspectives and are therefore visually related with a light blue rectangle that surrounds them. This rectangle also depicts the fact that these videos are to be annotated together, later on during the annotation procedure. In this way, 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.

The Video Explorer interface has been implemented in such a way that the user can click anywhere on a “hyper-video” and initialise an annotation session by opening the second component of the DC application, the Annotation Panel.

The Annotation Panel

The second component of the DC has as its main function to provide the user with the tools to add image and sound annotations to the video/videos already selected through the Video Explorer (Sevasti & Bouras, 2000). The main role of the annotation activity is to classify an episode so it could be kept in the different categories of the database. It also enables children to schematically express their immediate reactions to an episode.

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.

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 initialising a playback procedure and vice versa. In other words, the user is provided with the ability to interrupt the annotation procedure, to go back and find out how he/she has done so far, but during playback he/she can still pause the video reproduction and add annotations. We have called this “dynamic annotation.”

In the Annotation Panel environment, the video thumbnails are enlarged and occupy one, two or three corresponding playback screens (Figure 2). They are in fact not thumbnails anymore. They comprise small display screens of the hyper-video embedded in the Annotation Panel environment. These screens are used as canvases for the users to add their annotations.

Thus, the users can directly associate objects or actions displayed in the videos comprising a hyper-video, by placing above or right behind them the corresponding annotation (either facial expression or sound). Three simple video control buttons are provided to the user for controlling playback of the hyper-video being annotated. These 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. All buttons have the same effect on the video displays within the Annotation Panel interface environment, so that when for example the user presses the Play button, playback of all the three videos (in the case of Figure 2) starts simultaneously.

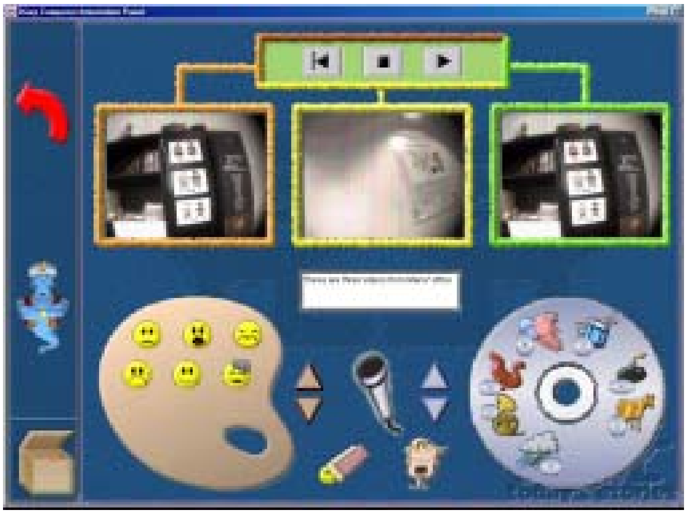

Figure 2. The annotation panel layout

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 while the second one (see bottom right of Figure 2) contains image annotations that are depicting sounds. The user is provided with the functionality of pausing the hyper-video video at any moment (by pressing the Stop button) and adding to it an unlimited number of image and sound annotations, to comment on the incident that has been recorded.

Apart from the predefined set of sound annotations, new sound annotations (e.g., spoken words, and sounds that express emotions) can also be inserted to the current hyper-video. The application in its current version also supports the feature of “scrolling palettes” for browsing through multiple annotation categories. Defining this dynamic set of annotations is a functionality provided by the Custom Annotations tool described later.

The Annotation Panel screen also offers users a text box for adding a brief text message (100 characters) as an annotation to the videos being annotated at any moment. Finally, on the top left corner of the Annotation Panel interface, a Back button exists that can be used for 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. At the moment when the users presses the Back button all new annotation additions and/or removals are recorded as final and the system’s database is updated accordingly.

Initialisation Applications

Initialisation applications are used for the insertion of initial values in the system’s database, so that they will be used by the DC application later. Three initialisation applications have been implemented: (a) the Teacher initialisation application, (b) the Administrator initialisation application, (c) and the Kid initialisation application, for teachers, technical people, and kids to be able to initialise and configure the DC application themselves. A detailed description of each initialisation application is provided in the following sections.

Teacher Initialisation Application

Teachers can use the Teacher initialisation application, to insert into the system’s database, information about the system’s entities. The Teacher initialisation application consists of the following modules: (a) the application frame which is the main module of the Teacher initialisation application and appears when the application starts, hosting all the other components of the application. (b) the “insert to database” modules that are responsible for inserting data to the system’s database and comprise individual panels and (c) the “delete and update” modules, which are responsible for deleting and updating data to system’s database. The user can access the latter from the delete and update panel, which provides a list with all the database data that are available for deleting or updating. The delete modules provide the capability for teachers to delete data from the system's database (if this is possible) with the use of database routines that the application frame module offers.

Database access is only provided through the application frame module to avoid having many database connections open at the same time. Thus, database access is kept centralised and simple.

Administrator Initialisation Application

Technical staff can use the Administrator initialisation application to insert into the system's database, information concerning for example, the multimedia file formats supported by the DC application. The Administrator initialisation application has the same architecture as the Teacher initialisation application and consists of the following modules: (a) application frame, (b) insert to database modules and (c) delete and update modules.

Kid Initialisation Application

The Kid initialisation application "recognises" a kid carrying an infrared beacon and displays on screen: (a) the kid's name in a fancy way, (b) the representative image of the kid (if it has already been inserted to the system's database by the kid himself or the teacher), and (c) the representative colour of the kid (if it has already been inserted to the system's database by the kid himself or the teacher). If one or both of the latest two entities (representative image of the kid and representative colour of the kid) has not been already inserted to the database or the kid wants to update/modify the existent entries, the Kid initialisation application provides each kid the functionality to: (a) choose an image that he/she would like to be represented by inside the DC environment and (b) choose a colour that he/she would like to be represented by inside the DC environment. Figure 3 shows the Kid initialisation application interface.

The Kid initialisation application consists of the following modules: (a) the application frame, which is the main module of the application and frame appears when the application starts, hosting all the other components of the Kid initialisation application; (b) the name panel that displays the name and the current image of the kid; (c) the "select image" panel, which provides an interface to each kid for him/her to select a new image and insert it to the system's database (with the use of database routines that the application frame module offers); (d) the “select colour” panel that provides an interface to each kid for him/her to select a new colour and insert it to the system’s database (with the use of database routines that the Application frame module offers); and finally (e) the colour Panel that displays the current colour of the kid.

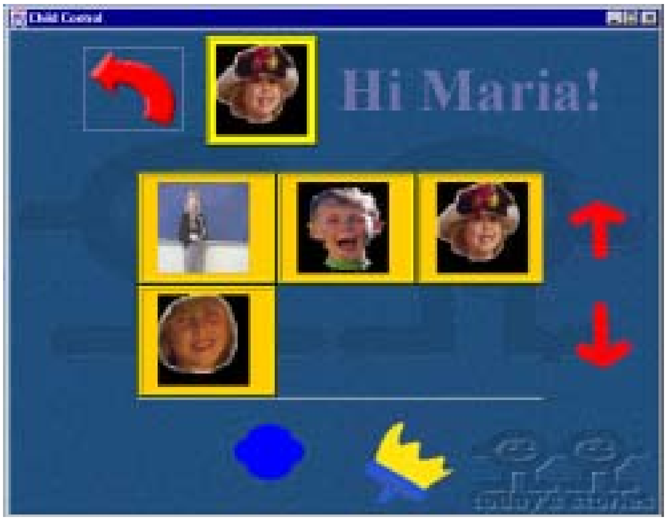

Figure 3. Kid initialisation application

The Custom Annotations application

The purpose that the Custom Annotations environment serves is that of allowing teachers, in cooperation with kids, to preview previously digitised material (images and sounds) and denote them either as image annotations or as sound annotations within the DC application.

The Custom Annotations environment is comprised mainly of the following components:

- the image palette, where the annotations of each image annotations’ category are displayed and the scrolling arrows of the image palette;
- the sound palette, where the annotations of each sound annotations’ category are displayed and the scrolling arrows of the sound palette;
- the scrolling panel, where all image files (*.gif or *.jpg files) existent in a system directory are displayed and the clipboard where images chosen

from the scrolling panel to become annotations in the palettes are temporarily placed; and

- the dustbin where annotations from the palettes and candidate annotations for the clipboard can be dragged-and-dropped to be deleted.

In Figure 4, these components are explicitly labelled. There are certain rules behind the operation of the Custom Annotations application. There are two kinds of annotations that users can define: image annotations and sound annotations. Image annotations can be displayed on the image palette (see Figure 4) while sound annotations are displayed on the sound palette (see Figure 4). At the same time, both image and sound annotations are organized in categories so for example, a set of image annotations that are displayed on the same image palette while scrolling through the image palette contents, belong to one image category.

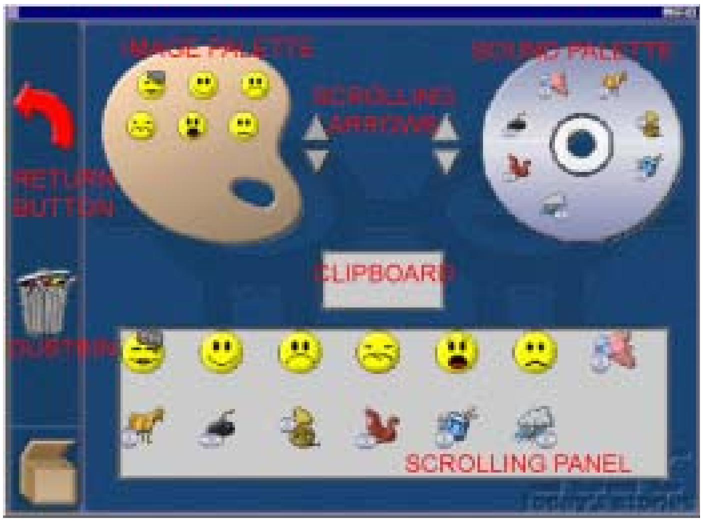

Figure 4. The components of the custom annotations environment

The set of annotations and categories that are currently existent every time the user terminates a session in the Custom Annotations environment, are the ones that can be used in the Annotation Panel of the Diary Composer. This means that the Custom Annotations environment also support automatic saving of the users' work.

Through the Custom Annotations environment, the users can browse through the existent set of annotation categories and their contents. The procedure of inserting digital material to the Custom Annotations environment is very simple and falls mainly in the responsibility of teachers. For such image files to be accessible by the Custom Annotations environment, users have to store them in a specific directory. In that way, all custom image files existing or recently stored in the aforementioned directory, appear in the Scrolling Panel of the Custom Annotations environment (see Figure 4) and can be used to create image and sound annotations.

Teachers, in cooperation with children, can also record digitised sounds and store them in a specific directory. In that way, all sound files existing or recently stored in the aforementioned directory, will be available later on, when creating a new sound annotation (or updating an old one), as a selection of sound for the user to choose from. Adding a new sound annotation also involves choosing a sound to associate the new annotation with from a window that pops-up, enabling users to preview sound files.

The user can choose to insert a new annotation to the current set of image or sound annotations either in an existent category or in the empty category (thus initiating a new images/sounds category). Furthermore, the user can remove one or more annotations from their categories one by one, by dragging and dropping them on the Dustbin. However, if the user tries to remove an annotation that has been attached to one of the videos of the Diary Composer system, he/she is not allowed to do so.

The Video Retrieval Application

To enable a continuous elaboration and reflection of the life episodes, their observation from different perspectives of time and children and sharing with others, it was also necessary to enable children to "revisit" distinguished video episodes selected according to certain criteria. This raised the need for a Video Retrieval tool within the DC environment, which is described in this section.

The Video retrieval application provides the capability for users to search into the DC system's database for videos according to various criteria and combinations of criteria. The Video retrieval application supports searching based on the following criteria (or any combination of them): (a) searching by Year, (b) searching by Month, (c) searching by Day of Week, (d) searching by Image or Sound Annotation, and (e) searching by Keyword. Figure 5 shows the Video retrieval application layout. The following paragraph describes the architecture and implementation of the application.

The Video retrieval application consists of the following modules: (a) the application frame, which is the main module of the application and appears when the application initiates, hosting all the other components of the application; (b) the search panel modules that provide the user with the capability to specify the search criteria; each of these modules consists of a panel to select and deselect search criteria. (3) The results panel module, which is responsible for presenting to the users the videos that match the selected criteria; for each selected video the system presents the video's thumbnail picture; and (d) the interface component that consists of the annotation movement module and the exit application module.

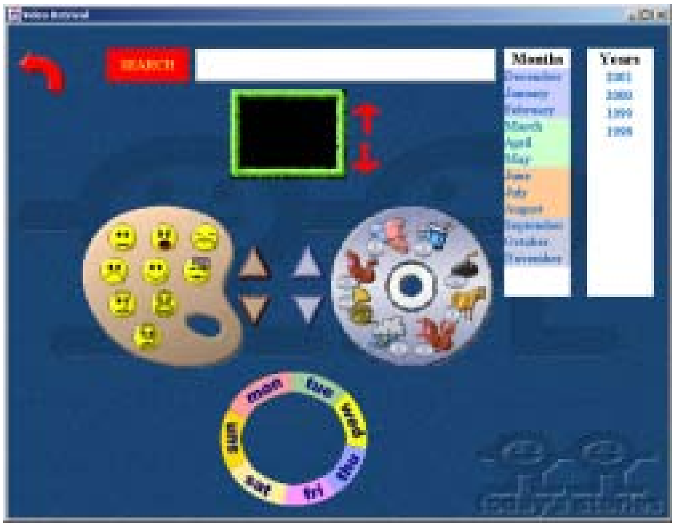

Figure 5. Video retrieval application

IMPLEMENTATION ISSUES

Annotation

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

Due to the lack of mature standards and to achieve acceptable performance and platform independence in application execution and video supported formats, an open and configurable annotation system was designed. 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, the limitations and performance issues arising from all the attempts to encode video and annotations together were bypassed.

All of the 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 earlier feasible.

The annotation procedure can be interrupted by the user at any time for him/her to be able to preview his/her annotations so far. The application then reads from those dynamic data structures 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/she wishes. This functionality is internally implemented by removing the correspondent entries from these data structures and reordering 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 in the system for future use. For the current version of the application, a proprietary scheme for storing all this information has been implemented.

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 [Sun Microsystems, 1999], 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.); and
- 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.

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 (Pantham, 1999; Piroumian, 1999) 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' behaviour 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 6.

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.

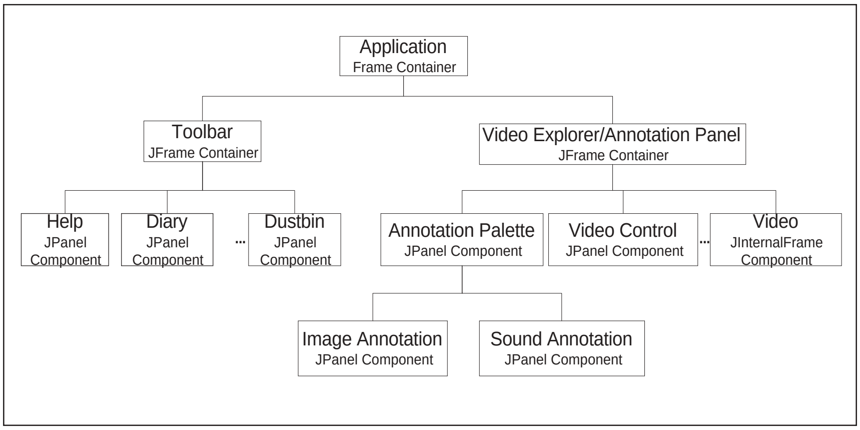

Figure 6. The multi-containment infrastructure

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 (Piroumian, 1999). 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 (Figure 7).

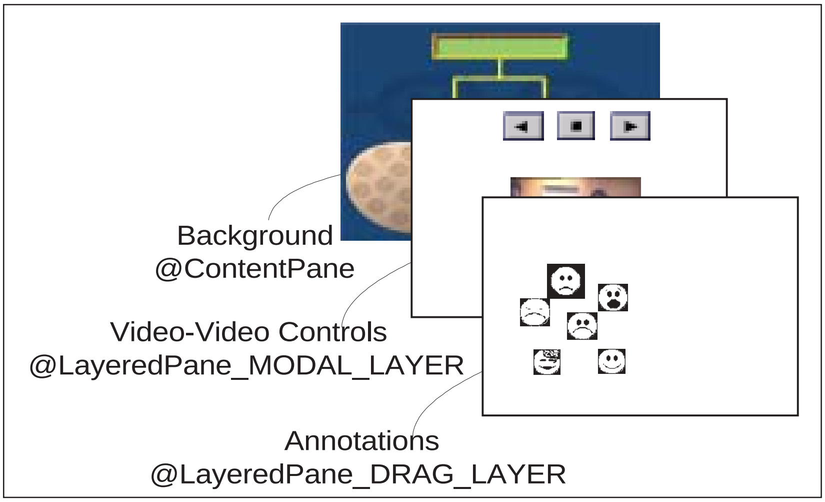

Figure 7. The multi-layer infrastructure

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 behaviour of annotations and the "Genie" object in the best possible way. Annotation and Genie objects' behaviour 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 behaviour was implemented over the mouse listening interface that the Java 2 SDK provides.

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 behaviour throughout the application runtime. In this section, how the JMF API was used for the DC implementation and the design choices made in order to achieve the desired functionality is presented.

According to Sun Microsystems (1998), 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 Carmo (1999), JMF does not build the functionality of constant media progression tracking into each media player. Based on this, 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 had to be implemented.

One of the dominant components for video/audio management within the DC was the Frame Positioning Control (FPC) interface of the JMF API (Carmo, 1999; Sun Microsystems, 1999). This interface was used for accessing the individual frames of each video file, a feature that is not built-in within JMF. 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.

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 the 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, a thread running in parallel to the video player/s had to be implemented.

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. These data are referred to 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+4 and AIFN>CVFN-4)

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 eight video frames (starting from four frames before the AIFN and finishing at four 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 methods that designated as master player in a group of two or three videos the one with the maximum duration had to be implemented. Also several experiments were made that resulted in the conclusion that equal frame rates among the members of a video group are ensuring best synchronization and performance behaviour.

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 being able to provide videos' fast forward and rewind functionality within the application in its current version. This issue is part of future work and might also be resolved in future versions of the JMF API.

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 article. A quantitative amount of experimenting took place to determine the appropriate video format characteristics that would allow qualitative simultaneous playback of three different video files in the Annotation Panel component. Results indicated as the only solution recommended for good performance was that of reducing the video files' frame rate to numbers less than or equal to 15 frames per second. According to the results of Ghinea and Thomas (1998), we are allowed to do so for the sake of playback quality, without putting at 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 that 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 synchronous 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.

Search Engine Implementation of Video Retrieval Application

The following expression describes the operation of Video retrieval application's search engine:

Search Result = Cat-1(Item-1 OR Item-2 OR … OR Item-n) AND Cat-2(Item-1 OR Item-2 OR … OR Item-n) AND … AND Cat-N(Item-1 OR Item-2 OR … OR Item-n)

Where:

- Cat-1, Cat-2, …, Cat-n are of the following categories: Year, Month, Day of Week, Annotation, Key word.
- Item-1, Item-2,…, Item-n are the selected items of the specified (Cat-x) category. For example if the specified category is the Days of Week category the items will be one or more from the following list: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday

The selection operation has been implemented as follows:

- Step 1—Start: All the available videos from the system's database and the selected videos are stored in a vector named Results
- Step 2—Refine Keywords: If the user has inserted a keyword as a search criterion, the system selects all the videos that match the given keyword (with the use of the following SQL command: select * from text where text like '%Keyword-1%' OR '%Keyword-2%' … OR '%Keyword-n%') and stores them to Keyword vector. The Results vector is updated to contain only the videos that previously appeared both to the Result vector and to Keywords vector.
- Step 3—Refine Annotations: If the user has selected annotations as search criteria, the system selects all the videos that match the given annotations (with the use of the following SQL command: select * from video_annotations where item_id in (annotation-1, annotation-2,…, an-

notation-n) and stores them to the Annotations vector. The Results vector is updated to contain only the videos that previously appeared both to the Result Vector and to the Annotations Vector.

- Step 4—Refine Years: If the user has selected years as search criteria, the Results vector is updated to contain the videos that appear to the Results vector and were shot during the selected years.
- Step 5—Refine Months: If the user has selected months as search criteria, the Results vector is updated to contain the videos that previously appeared to the Results vector and were shot during the selected months.
- Step 6—Refine Days of Week: If the user has selected days of week as search criteria, the Results vector is updated to contain the videos that previously appeared to the Results vector and were shot during the selected days of week.
- Step 7—Final Result: The Result vector contains the videos that match all the selected search criteria.

FUTURE WORK—CONCLUSIONS

The DC application presented in this article is comprised of the first fully functional version of a tool that has been gradually enhanced with functionality and improved according to the feedback that has been provided from trials in school environments within the "Today's Stories" project time plan and the pedagogical analysis of the application's use and nature.

Apart from adding functionality to the existent tool (such as making the DC environment accessible over the Internet), 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 to 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 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 MPEG-7 standard will contribute significantly towards this direction.

References

Aviram, A. (1993). Personal autonomy and the flexible school. *International Review of Education, 39*(5), 419-433. The Netherlands: Kluwer Academic.

Baecker, R., Rosenthal, AJ., Friedlander, N., Smith, E., & Cohen, A. (1996). A multimedia system for authoring motion pictures. In *Proceedings of the 4th ACM International Multimedia Conference,* (pp. 31-42). Boston, MA: ACM Press.

Benz, H., Bessler, S., Fischer, S., Hager, M., & Mecklenburg, R. (1997a). DIANE: A multimedia annotation system. In *Proceedings of the Second European Conference on Multimedia Applications, Services and Techniques (ECMAST'97*). Milan, Italy.

Benz, H., Fischer, S., Mecklenburg, R., & Dermler, G. (1997b). DIANE - Hypermedia documents in a distributed annotation environment. In *Proceedings of the Conference on Hypertext* - *Information Retrieval* - *Multimedia (HIM'97)*. Dortmund, Germany.

Bouras, C., Kapoulas, V., Konidaris, A., Ramahlo, M., Sevasti, A., & Van de Velde, W. (2000). Diary composer: Supporting reflection on past events for young children. In *Proceedings of World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Montréal, Canada (pp. 105-110). Charlottesville, VA: Association for the Advancement of Computing in Education.

Carmo, L.D. (1999). *Core Java media framework*, Upper Saddle River, NJ: Prentice Hall.

Chiueh, T., Mitra, T., Neogi, A., & Yang, CK. (1998). Zodiac: A history interactive video authoring system. In *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98),* (pp. 435-443). Bristol, UK: ACM Press.

EBU/SMPTE-Task Force for Harmonized Standards for the Exchange of Programme Material as Bitstreams (1998). *Final report: Analyses and results*. Author.

Geissler, J. (1995). Surfing the movie space: Advanced navigation in movie-only hypermedia. In *Proceedings of the 3rd ACM International Multimedia Conference (Multimedia'95),* (pp. 391-400). San Francisco, CA: ACM Press.

Ghinea, G., & Thomas, JP. (1998). QoS impact on user perception and understanding of multimedia video clips. In *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98),* (pp. 49-54). Bristol, UK: ACM Press.

MPEG-7 Applications Document v.9 (1999). In A. Lindsay (Ed.), *ISO/IEC JTC1/SC29/WG11/N2861*. Vancouver, Canada: International Organisation For Standardisation Requirements Group.

Pantham, S. (1999). *Pure JFC swing*, Indianapolis, IN: Sams Publishing.

Piroumian, V. (1999). *Java GUI development: The authoritative solution*, Indianapolis, IN: Sams Publishing.

Ponceleon, D., Srinivasan, S., Amir, A., Petkovic, D., & Diklic, D. (1998). Key to effective video retrieval: Effective cataloging and browsing. In *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98),* (pp. 99-107). Bristol, UK: ACM Press.

Santos, C.A.S., Soares, L.F.G., de Souza, G.L., & Courtiat, J.P. (1998). Design methodology for formal validation of hypermedia documents. In *Proceedings of the 6th ACM International Multimedia Conference (Multimedia'98),* (pp. 39-48). Bristol, UK: ACM Press.

Sevasti, A., & Bouras, C. (2000). Using Java to implement a multimedia annotation environment for young children. In *Proceedings of the 8th ACM International Multimedia Conference,* (pp. 187-194). Los Angeles, CA: ACM Press.

Sun Microsystems Inc. (1999). *Java media framework API guide*. Retrieved March 2002 from: http://java.sun.com/products/java-media/jmf/2.1.1/guide/

Sun Microsystems Inc., Silicon Graphics Inc. & Intel Corporation (1998). *Java media players*. Retrieved March 2002 from: http://java.sun.com/products/java-media/jmf/1.0/guide/index.html

Today's Stories i3 –ESE (Long Term Research Task 4.4) Project Nr. 29312 (2001) [Online]. Available: http://stories.starlab.org/about.htm

Yoo, S. (1995). *Multimedia authoring/scripting.* Course seminar. Retrieved March 2001 from: http://mmlab.snu.ac.kr/course/mmseminar/temp/YsPres.html