Joachim Griesbaum, Thomas Mandl, Christa Womser-Hacker (Hrsg.)

Information und Wissen: global, sozial und frei?

Proceedings des 12. Internationalen Symposiums für Informationswissenschaft (ISI 2011)

Hildesheim, 9.—11. März 2011
Effects of real, media and presentation time in annotated video

Peter Schultes¹, Franz Lehner², Harald Kosch³

University of Passau – Innstrasse 43, Passau
1 Business Administration II – schult16@stud.uni-passau.de
2 Business Administration II – franz.lehner@uni-passau.de
3 Distributed Informationsystems – harald.kosch@uni-passau.de

Abstract

In recent years, annotated video became a major factor of our multimedia world. As we can see using the example of YouTube, annotating an existing video with graphical objects seems to get very popular these days. But a lot of technical problems are not solved yet or are still even not identified. One of these issues is dynamic behavior of (graphical) annotations, which current platforms generally do not support at all (for instance YouTube). This work deals with a question that is of inherent importance for dynamism: how can we synchronize media, real and presentation time in interactive video? Current platforms typically support media time dependencies, but discount the fact that user interactivity takes place at presentation time. Our approach of handling different time systems enables interactive video to expand to a much greater field of application.

1 Introduction

Just a few years ago, video was a linear presentation medium. We generally interacted more with our remote control of our video players than with the media itself. Then in the mid-nineties the DVD came up and interactive multimedia presentations emerged: additional material, alternative endings, audio comments and so on. After suitable technologies like Adobe Flash and Microsoft Silverlight took up the internet at the beginning of the twentieth century, clickable video appeared. Illustration-, advertising- and entertainment-
videos are enriched with interactive features for the viewers since then. Nowadays – in times of web 2.0 – we can see an ongoing development: Video users themselves publish videos, annotate existing material and even enrich the videos with interactive features. Especially due to YouTube (c.f. [YouTube, 2009]) it became very popular for users to add graphical shapes, interactive hotspots or links to other media into an existing video. As we can see, the traditional observer role of video spectators seems to break up more and more. Interactive media on the one hand and user generated content on the other hand are the two main trends in the area of internet video. However, the possibilities for user generated video annotations are still limited. Of course we merely can do everything we want with a video by using programming technologies like Adobe Flash. But the majority of the video community cannot handle these technologies, because they require a lot of technical knowhow and experience. And YouTube annotations are certainly not the best answer either: only three different graphical shapes, as good as no dynamic and interactive behavior – this appears more to be the first step but not the final state.

The main focus of our current work is to develop an annotation tool for online videos that everybody can easily use. The primary application field of our video annotation tool is eLearning. We would like to give our students the possibility to add additional material, references, notes etc. into their (customized) online lecture videos and share them within a peer group or a public domain. One of our research aspects here is to explore how far video is a suitable base media for user generated content. The supported video annotations include all kinds of graphical information (not metadata) like text, shapes, freehand drawings, audio, images or even video. Currently we are still in realization stage but we already identified some key issues of user generated video annotations. One issue is dynamic video annotations. Dynamism in this context contains changes of visual properties of annotation elements during video presentation. Visual properties are for example the boundary or background color of an element. A property change can be evoked by either a timer (e.g. media time progress) or the user, who interacts with an annotation element.

In this article we worked out the main fields of problems that came up when realizing dynamic video annotations. The main challenge here was to bring media time dynamics and real time interactivity together. In the next sections we explain technical issues and semantic concepts on how to achieve dynamic and interactive video annotations in a consistent and generic way.
2 Related work

In recent years, the number of projects dealing with the topic of interactive video enormously increased. The following list gives a brief overview of the priorities in latest research and industrial projects including the main features – making no claims of being complete.

non-linear video

The Fraunhofer Institute has presented their “nonlinear-video” project at the cebit 2010. By own accounts, nonlinear-video enables us to “re-experience the content of moving pictures”. The main feature here is a whole video environment. The environment provides us with the possibility to interact with annotated video objects and retrieve sensitive information about the video content (c.f. [Fraunhofer, 2010]). Additional to the desktop presentation, developers at the Fraunhofer Institute are also working on a hardware set-top box. This could bring interactive television into our living rooms.

ADIVI

The ADIVI project (cf. [InnoTeamS, 2009]) – which is also a commercial software solution – focuses on additional video annotations. The system consists of fixed video and annotation areas. The content of the annotation area is determined by sensitive regions in the video. If the viewer clicks on a specific region, annotation area shows up all kind of additional media resources (text, images, videos, URLs).

Microsoft Video Hyperlinks

Microsoft adCenter Labs published Video Hyperlinks (see [Microsoft, 2008]) in 2008. Here, hotspots can be placed in an existing video to mark regions of interest with rectangular frames. The hotspots appear at certain media time points and change their bounding according to the background objects. After the users clicks inside the rectangle, additional information is shown.

SIVA Suite

SIVA Suite is one of the latest research projects of the University of Passau. Basically SIVA is a comprehensive collection of software tools, which can be used to create and play interactive videos. As well as the previous projects SIVA provides synchronous presentation of all kinds of media annotations
3 Fundamentals

In a previous project, we already identified three basic components that make up a dynamic video-annotation system (c.f. [Schultes (1), 2010]) in addition to the video player: first of all, the annotation elements which appear upon the actual video content. At second, the environment which manages the containing elements. And third, the abstract screenplay which defines interactivity and dynamics during the presentation. These essential concepts are summed up in the next sections, because they serve as the basis for all further considerations.

3.1 Elements and Environment

The focus of our work lies on visual annotation elements like shapes, text, images, videos, or interactive elements. All this video annotation elements are instances of a common interface. The exact nature of this interface should not be discussed further at this point (you can find more details in [Schultes (1), 2010]). For further explanations we can simply assume the following prerequisite: Each annotation element provides the same generic mechanism to implement element specific behavior. We can also control and manipulate the state of an element via the interface without having detailed information about the particular element. The controlling of all annotation elements is done by the runtime environment of the collaborative video player/editor. So the annotation environment is the connection between video and user generated content and has three major functions: The first task is managing the graphical annotation context. The second task is to interpret the abstract screenplay and so insert/hide the annotation elements at certain time points. And at last the environment is responsible for forwarding all changes and events – for instance user inputs – to the annotation elements (see next section).
3.2 Events

The annotation environment has to react on multiple kinds of events and state changes:

- the ongoing position in video, thus the current media time
- timing events from the system clock
- system events from user input devices (e.g. mouse clicks)
- element specific events (e.g. selection of a button shape)

After the annotation environment noticed an event or an element state change, all detail information is packed into a standard event object. This generally consists of an event type identifier, an explicit source element and arbitrary detail objects (e.g. which mouse button was pressed etc). This proceeding is required, because the common element interface can only deal with one particular (generic) event object. The event object must then be interpreted by all involved elements. An element is usually involved in an event, if it represents a user interface event, which occurred in the graphical bounds of the element. The next step is to inform appropriate event handlers about the occurrence of the event (see next section).

3.3 Actions and Dynamics

Dynamics and actions are defined in the abstract annotation screenplay. The screenplay has to be created by the author of an interactive video prior to or during its presentation – in our case the screenplay is collaboratively created by the users of an internet community. Its evaluation is done by the environment at video runtime. Realization issues of the screenplay are not part of this work – you can read further details about our “event tree”-approach which deals with the question on how users can define and manage complex screenplays graphically in [Schultes (2), 2010]. Basically, the screenplay holds an amount of actions which will be executed in response to particular events. The connection of runtime event and predefined action is called an event handler. An event handler specifies its trigger event by an appropriate event prototype. An example event handler could possibly be something like: 

After selection of button ‘xy’ annotation element ‘yz’ should set its background colour to blue.
Therefore, the event prototype would map to all runtime events whose type is “selection” and arise from element “xy”. The related action would change the background property of element “yz”.

In this context, an action is a container for an arbitrary amount of instructions. Each instruction always references a target element and defines an explicit command directly affecting this target. A command can easily be realized via the generic element interface (see section 3.1). Usually instructions (and so actions) change the state of their target elements which can lead to the occurrence of new element events. Further to the event handlers, an abstract screenplay contains all appearing and disappearing time points of every visual annotation element. This is essential because the elements usually do not remain visible across the entire video.

4 Real time and media time

The annotation environment initiates the execution of certain actions at particular time points. These time points always have an explicit timestamp depending on the system clock of the underlying hardware system. By default this timestamp corresponds to a real time point in the existing time continuum. Since video is a time based medium, each real time execution point can be mapped to a corresponding media time point. Media time points are in contrast to real time points singletons and so recoverable. For example, viewers can jump back and forth in media time by adjusting the current position in the video. This certainly does not apply to real time unless time machines get affordable in the near future. The presence of several execution time points arise the question of which time point is decisive for the execution time of an action.

4.1 Media time actions

Media time events primarily include frame updates in the video player but also derived events like “element xy is now shown” (since media time points which change the hide/show states of annotation elements are explicitly listed in the screenplay, these events can be treated like “real” media time events). In order to give the impression that the annotation elements are an integral
part of the underlying video they strictly have to behave synchronously with the video content. Let’s consider the screenplay contains an event handler which changes the background color of an annotation shape at media time 00:05:15. It seems obvious that the background color must be reset to its original value in case of a media time setback, to keep synchronous with the video. This demand is getting even more obvious, if we take a look at moving elements (which often appear in connection with hotspots). Let’s assume that the screenplay contains several different location requests targeting one particular annotation shape at certain media time points. As a result, the element acts synchronously with a background object in the video. Considering that, a media time step back inevitably forces a location update, unless user generated content and video lose their synchronicity. Figure 1 shows this circumstance:

Figure 1:
Loss of synchronism due to moving elements, when ignoring a media time step back.

So, in case of a media time backward jump, each annotation element has to restore the most recent valid state before the target media time. Therefore, the annotation environment has to manifest the state of the target element before executing a media time action. This is the only way to ensure proper element states in case of future media time jumps. In terms of software engineering this approach is called Memento (c.f. [Gamma, 1995]) and is used in similar circumstances: realizing undo-redo mechanism in graphical user interfaces.

As we have seen, backward steps require a special treatment. But how to handle forward steps? Of course we would lose synchronicity of annotation
Effects of real, media and presentation time in annotated video

elements and video, if we ignored forward steps. For example, if the viewer (in respect of the scenario in figure 1) jumped from media time point 2 to 7, the location update would fail to appear. The required action will only be executed on media time event 6, which was skipped due to the forward jump. Thus, the annotation environment has to predetermined the skipped state transitions and manifest them at the given media time points respectively. The environment can achieve this easily by using the given event prototypes details in the abstract screenplay. Forward calculation is done in three sequential steps:
- Calculation of all media time actions which occur in the skipped time span.
- Sorting of the selected actions according to their execution time points.
- Sequential execution of the actions and state manifestation of affected elements according to execution time points.

4.2 Real time actions

In contrast to media time actions, real time actions do not depend on the video presentation at all. Real time actions change the state of an annotation element at a certain time point triggered by the system clock. An example real time event would be: “12:00:35 on 01-05-2010”. Since it is not possible to reset the real time, we do not have to care about element state manifestation and so on and only have to evaluate the necessary system clock events. However, the environment must ensure the correct initial state of all elements at the beginning of the video presentation: if an event handler updates the colour of an annotation shape at 12:00:35 and presentation starts at 13:00:00, the environment should catch up on the missed real time event and update the element before presentation starts. This guarantees consistent behaviour of user generated annotations, no matter at which time of day the presentation starts.

In general, real time actions as described here would not make much sense for normal use cases. They are more an issue for commercials on top of the video, which are dynamically shown at particular times of day. But certainly, real time actions only play a minor role. However, we should consider the idea as useful for element specific behaviour. Let’s take a look at an interactive door-shape: a mouse click could trigger a visual opening- or closing-animation. Animations are usually real time based, and must continue
even if the user pauses the video. So here we do not have absolute time-stamps but relative ones, dependent on the system timestamp of the trigger event.

### 4.3 Presentation time actions

The previous considerations dealt with actions that are triggered by any sort of time events. All further events belong to the presentation time category: for one thing there are events fired by input devices (e.g. mouse pointer movements/clicks or keystrokes ...), for another thing element specific events (e.g. “selection performed” in an interactive multiple choice element). These events occur in the annotation layer, thus outside the actual video context, but mostly depend on the video content. For instance, a mouse click on a video object always relates to current media time – the viewer would probably not have clicked on this particular location, if the video content had been a differed one. This is also transferable to user interactions with the annotation layer: annotation elements are always closely linked to the video content and media time due to their dis-/appearing (media) time points. This raises the question of whether presentation time actions have to be rolled back as well as media time actions in case of backward jumps. Forward calculations do not make sense at all, since the environment cannot estimate possible interaction events during the skipped time span (which might have occurred in a continuous presentation). But if the annotation environment manifested the element states in case of presentation time actions, too, they could be restored after a backward jump. The following example scenario should help us to understand this problem:

In an interactive video project, viewers should receive additional information to objects they selected with the mouse pointer. Therefore we have a text field, which always stays visible during the entire video presentation. Furthermore, annotation shapes appear and disappear at specific media-time points. These elements act as hotspots for corresponding objects in the video. After the viewer clicks on such a hotspot element, the text field presents additional information about the enclosed video object.

Let’s assume a viewer has watched the first part of the interactive video and has already selected some hotspot elements. Now he wants to set back media time to watch the last scene again. This leads us to two different posi-
tions whether the content of the text field has to be reset or not (after the backward jump):

- **Position 1:** Mouse click occurs at real time, but the corresponding action is executed at media time.
  
  The environment must reset the text field after a media time setback, because its content changed through (video-) content sensitive actions. This ensures synchronicity with the video.

- **Position 2:** Mouse click occurs at real time and corresponding action is executed in real time context, too.
  
  The content of the text field is still valid, because the last user selection took place before the media time setback. So no rollback is required. This ensures correct real time behaviour.

One consequence of position two is that the text field may contain information about a video object, which has not been shown yet. This could lead to uncertainty among the viewers. But position one also shows a similar weakness: after the media time setback, all elements states suit to the current video content. If the user performs a forward jump right after the setback, the elements cannot restore their previous state again (since forward calculation is generally not possible – even if the target time point was recently shown).

So which of the positions is the “right” one? In general, we cannot predict the correct handling of media time changes, if we do not know the particular use case. If there is a strong connection between video content and user action (as in the example scenario), viewpoint one seems to be the best suitable. If not – for instance in case of element events like button selections etc. – viewpoint two would be a better choice. Due to the fact that the annotation system cannot calculate the correct handling, the best solution here seems to be a preliminary commitment in the abstract screenplay. The author of an interactive video should have the possibility to specify the desired handling for each presentation time action by himself.

### 4.4 Special cases

One special case needs to be discussed further at this point: media time actions can change the state of an element. This can evoke new element events at the target element(s). As far as one or more of these (possible) events are connected with other actions, “action chains” arise. Regardless of whether position one or two is taken for media time handling, action chains whose
start event is a media time event, have to be completely rolled back. So the context of action chains is always determined through the start event. This dependency seems to be obvious, since there would not be any further events without the start event. But there is one exception: if the implementation of a chained action adjusts the current video position, the context switches at this chain index. So any further action of the chain will require element state manifestations. This may lead to pretty obscure behaviour, which we have not considered so far. It could even be possible that a chained action triggers another action chain. A lot of problems with action chains are not considered so far and will be part of future work.

5 Conclusions and future work

In this work we introduced the basic concepts of handling dynamics and interactivity of graphical video annotations. As we have seen, different action types may require a different treatment for media time jumps. In case of presentation time actions the environment even cannot determine, whether a roll back is necessary or not. So here we need information from the author on how to treat each particular presentation time action. Due to the results of our work, we were able to implement the logic for handling dynamics in annotated interactive videos.

In our future work we will concentrate on unresolved problems when media time affects video annotations (for example the above mentioned action chains). In the next step we will finish up implementing our annotation tool for internet videos. We expect a lot of problems when realizing collaborative viewing and editing of annotated videos, because this area is still only insufficiently explored. We will then integrate the final collaborative video annotation player in our online lecture courses. After having sufficient user content we will launch a study about the quality and different types of the generated user content. This should help us to get a sustainable estimation on how far video is a suitable media for user generated content.
References


E. Gamma, R. Helm, R. Johnson, J. Vlissides (1995): Elements of Reusable Object-Oriented Software. Amsterdam: Addison-Wesley Longman, 1995

http://www.adivi.net/Materialien/ADIVI_3_help_en.pdf (Retrieved 03.01.2011)


Microsoft adCenter Labs (2008): Video and Interactive.
http://adlab.msn.com/Video-Hyperlink/ (Retrieved 03.01.2011 from)


http://www.youtube.com/t/annotations_about (Retrieved 03.01.2011 from)