AcroTeX.Net

The **rmannot** Package
Demonstrating Navigation Cue Points

D. P. Story

A source file containing the example presented in this article is attached. Open the attachments panel (by clicking on the red link) and save to your local computer. There is a link on the AeB Blog page to download the FLV file (`sample.flv`) used in this article. The source code attached to this document has sample code for embedding the FLV rather than streaming it, as this article does.

1. About Cue Points

When the rmannot package was first published in 2008, it had support for cue points, but unfortunately, I could never make it work! After a couple of years, I came back to the package to pursue the problem. As it turned out, there was a typo in the rmannot package, something simple; once correct, cue points started working! The use of cue points, therefore, requires version v1.0 (dated 2010/09/24 or later) for the cue points to work correctly.

This is the first article in a series about cue points. The videos used to demonstrate cue points are on the www.math.uakron.edu server. The FLVs are available through a separate download.

A brief article on cue points may be found at Cue points for FLV and F4V video files. To summarize, cue points are encoded on an FLV when the video is converted to FLV using some utility, such as the **Adobe Media Encoder CS5**. You cannot create cue points on an already existent FLV, it must be done during the encoding process from another video format.

There are two types of *navigation* (demonstrated in this article), and *event*. Both types embed cue points in the video file. The usefulness of cue points is that you can associate ActionScript (or JavaScript, in the case of PDF) to execute at each cue point. This is demonstrated in this article, and in others to follow.

2. Navigation Cue Points

Navigation cue points are used for navigation and seeking, and to trigger ActionScript methods when the cue point is reached. Embedding a navigation cue point inserts a keyframe at that point in the video to enable viewers to seek to that place in the video.

The Acrobat version of navigation cue points allows us to execute JavaScript on PDF side instead of ActionScript on the SWF side. (The FLV is played by the embedded VideoPlayer.swf player that ships with Acrobat.)

The movie, seen on the next page was created with the following code,

```
\resizebox{.67\linewidth}{!}{%
    \rmAnnot[url,poster=aebmovie_poster,
        cuepoints={\myCuePoints}]{320bp}{240bp}{sample}}\\[6bp]%
    \textField[\BC{}\Q{1}\Ff{\FfReadOnly}
        ]{txtCues}{.67\linewidth}{11bp}
```

We use the url key to signal to \rmAnnot that the video file is external, on the web. The new part is the inclusion of the cuepoints key-value pair; in the code above, see cuepoints={\myCuePoints}. The \myCuePoints command is user-defined in the preamble (or anywhere before first use). For the sample.flv video, which is the video to be played in the rich media annotation for this document, \myCuePoints are defined as follows:

```
\def\myCuePoints{%
  {type=nav,name=Chapter1,time=0,action={\wrtTxt{Boy rides bike}}},%
  {type=nav,name=Chapter2,time=1900,action={\wrtTxt{Boy slides down}}},%
```

```
    {type=nav,name=Chapter3,time=5200,action={\wrtTxt{Boy crawls through tunnel}}},%
    {type=nav,name=Chapter4,time=6800,action={\wrtTxt{Boy spins cubes}}},%
    {type=nav,name=Chapter5,time=9100,action={\wrtTxt{Boy runs through playground}}},%
    {type=nav,name=Chapter6,time=12200,action={%
        \wrtTxt{End of movie, bye-bye, boy};
        var timeout=app.setTimeOut('\wrtTxt{}',2000);
    }}
}
```

The value of the `cuepoints` key is a *comma-delimited* list of key-value pairs. Note the comment characters (%) after the comma of each cue point. It is important to have the comment character because of the crude way of parsing comma-delimited list, this may be fixed in the future.

The original movie, `sample.mov`, was converted to a FLV video using the **Adobe Media Encoder**, and at that time the cue points were defined. The encoder has a feature for saving a copy of the cue point data as an XML file, it is this file that I used to set the cue point data in the `\myCuePoints` command.

Let's look at one of these cue points:

```
    {type=nav,name=Chapter2,time=1900,action={\wrtTxt{Boy slides down}}},%
```

The keys recognized by the `rmannot` package are `type`, `name`, `time`, and `actions`. The value of `type` may be either `nav` (for navigation) or `event`. Each cue point, occurring at a specific time, is assigned a name. The name is given when the file is encoded; the name and time data is written to an XML file for later use. The value of the `actions` key is JavaScript code that is executed when the cue point is reached.

Before continuing with this discussion, take a look at the example. Click on the rich media annotation below.

AcroTeX

Movies

D. P. Story
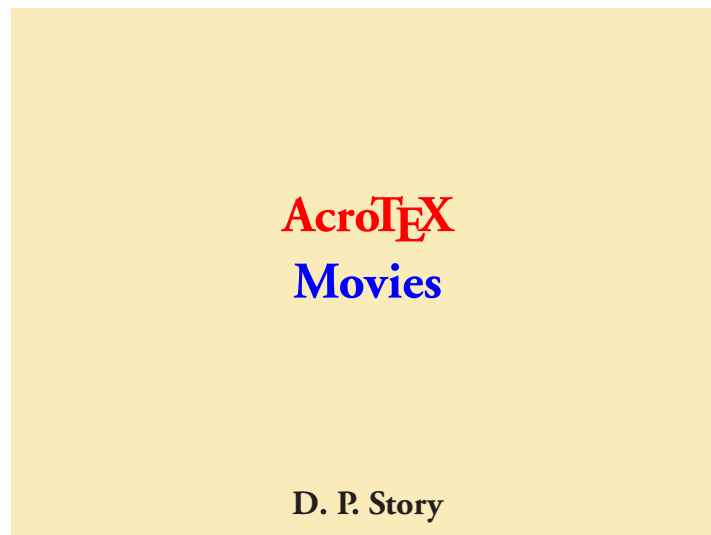
The action for each cue point is simple, a message is written to the text field directly beneath the movie. The action, `action={\wrtTxt{Boy slides down}`, for example, writes the message to the field when the boy slides down the slide. The command `\wrtTxt` is a convenience command defined as

```
\newcommand{\wrtTxt}[1]{this.getField("txtCues").value="#1"}
```

The cue points are of a navigation type. As you play the movie, you can pause it, and move to the next (navigation) cue point by pressing the "next cue point button" on the movie's control bar (the fourth button from the left), or you can press the "previous cue point button" (the third button from the left). In this way, you can move from "chapter to chapter" in the movie. The text field shows chapter titles.

2.1. Final Notes

The action for the last cue point has been embellished a bit.

```
{type=nav,name=Chapter6,time=12200,action={%
    \wrtTxt{End of movie, bye-bye, boy};
    var timeout=app.setTimeOut('\wrtTxt{}',2000);
}}
```

At the end of the movie, the last cue point is encountered. A heart-warming message is first written, then the `app.setTimeOut()` method is used to clear the text field after two seconds.

One of the many deficiencies of RMA (rich media annotations) is the lack of events. There should be an activation event, an event that fires when the RMA is activated. With the activation event, we could executed some startup JavaScript. There should also be a deactivation event, where we can execute JavaScript when the RMA is deactivated. For example, instead of using `app.setTimeOut()` to clear the field after 2 seconds (2000 milliseconds), we could clear the field when the RMA is deactivated, either by the user by right-clicking on the RMA, or automatically when the use moves to the next (or previous) page.

Another glitch that I see is that when you view the movie in full screen (right-click and choose Full Screen Multimedia after the RMA is activated) the movie drops out of full screen mode each time a cue point is encountered. This may be by design; recall watching movies or t.v. shows online through a browser, you drop out of full screen to watch a commercial.

Now, I simply must get back to my retirement! DS